Teil VI

Source-Code

Anhang I

Korpora – Code

I.1 Korpus-Verwaltung

Für das automatische Erstellen der `index.xml` Dateien wurden viele kleine Skripte geschrieben, von denen stellvertretend hier nur eines gezeigt wird.

I.1.1 koksUtil.tcl

Hilfsroutinen

```
# utility routines for the KOKS project
# Arno Erpenbeck, 06.01.2001

package provide koksUtil 1.0

namespace eval koksUtil {
    namespace export openFile closeFile
}

# open a file specified by the name for reading
proc koksUtil::openFile {fnm {mode r}} {
    if {[catch {set in [open $fnm $mode] } err]} {
        puts stderr "Error opening index file: $err"
        exit 1
    }
    return $in
}

# close the file specified by the handle
proc koksUtil::closeFile {handle} {
    if {[catch {close $handle} err]} {
        puts stderr "Error closing index file: $err"
    }
}
```

I.1.2 denews-add.tcl

`index.xml` für DE-News anlegen

```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh8.3 "$0" "$@"

```

```
# $Header: /opt/cvs/bin/denews-add.tcl,v 1.1 2001/10/21 15:23:19 koks Exp $

# find all de-news text files and create index.xml
# KOKS project
# Arno Erpenbeck, 23.02.2001

# load packages
lappend auto_path /home/koks/share/tcl
package require koksUtil

set fnm "index.xml"
set verbose 0

# check command line args
if $argc {
    if {![string compare "-v" [lindex $argv 0]]} {
        set verbose 1
    }
}

# traverse all subdirectories
foreach i [list 1996 1997 1998 1999 2000] {
    foreach j [list 01 02 03 04 05 06 07 08 09 10 11 12] {
        if $verbose {
            puts -nonewline "$i/$j: "
        }

        # find text files in current dir
        set files [glob -nocomplain $i/$j/*.txt]
        if {![llength $files]} {
            if $verbose {
                puts "empty"
            }
            continue
        }

        # get day, month, year from filename
        set dummy [split [lindex $files end] -]
        set year [lindex $dummy 2]
        set month [lindex $dummy 3]
        set max [lindex [split [lindex $dummy 4] .] 0]

        # create index.xml
        set out [koksUtil::openFile $i/$j/$fnm w]

        # write header
        puts $out "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>"
        puts $out "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">"
        set foo [exec date "+%T %d.%m.%Y"]
        puts $out "<!-- generated: $foo -->"
        puts $out "<mapping>"
        koksUtil::closeFile $out

        # create xml entry for all files
        for {set day 1} {$day <= $max} {incr day} {
            # use "01" instead of "1"
            if {$day <= 9} {
                set realday "0$day"
            } else {
                set realday $day
            }
```

```
            exec idx-add.tcl -x -p -l de de-news-$year-$month-$realday.de.txt plain -p -l en de-news-$year-$mont
        }

        # write footer
        set out [koksUtil::openFile $i/$j/$fnm a]
        puts $out "</mapping>"
        koksUtil::closeFile $out

        if $verbose {
            puts "ok"
        }
    }
}
```

I.2 Korpus-Browser

GUI für Korpus-Navigation (s. 3.6.2)

```
#!/usr/local/bin/ruby

require "gtk"
require "tempfile"
require "xmltreebuilder"
require "xmltreevisitor"

# $Header: /opt/cvs/bin/kbrowser.rb,v 1.9 2001/08/13 16:21:43 koks-cvs Exp $

# browse index.xml files
# KOKS project
# Arno Erpenbeck, 30.07.2001

# TODO
# - progressbar for long operations
# - fix encoding problems
# - new xpms for toolbar
# - make preproc.py call possible

# define paths for external viewer
$ps_viewer = "/usr/X11R6/bin/gv"
$pdf_viewer = "/usr/X11R6/bin/acroread"
$html_viewer = "/usr/X11R6/bin/netscape"
$preproc_cmd = "/home/aerpenbe/bin/preproc.py"

# define xpms for tree display
$book_open_xpm = [
  "16 16 4 1",
  "       c None s None",
  ".      c black",
  "X      c #808080",
  "o      c white",
  "                ",
  "   ..           ",
  " .Xo.    ...    ",
  " .Xoo. ..oo.    ",
  " .Xooo.Xooo...  ",
  " .Xooo.oooo.X.  ",
  " .Xooo.Xooo.X.  ",
  " .Xooo.oooo.X.  ",
  " .Xooo.Xooo.X.  ",
  " .Xooo.oooo.X.  ",
  "  .Xoo.Xoo..X.  ",
```

```
    "   .Xo.o..ooX.  ",
    "    .X..XXXXX.  ",
    "    ..X.......  ",
    "     ..         ",
    "                " ]

$book_closed_xpm = [
    "16 16 6 1",
    "       c None s None",
    ".      c black",
    "X      c red",
    "o      c yellow",
    "O      c #808080",
    "#      c white",
    "                ",
    "       ..       ",
    "     ..XX.      ",
    "   ..XXXXX.     ",
    " ..XXXXXXX.     ",
    ".ooXXXXXXXX.    ",
    "..ooXXXXXXXX.   ",
    ".X.ooXXXXXXXX.  ",
    ".XX.ooXXXXXX..  ",
    " .XX.ooXXX..#O  ",
    "  .XX.oo..##OO. ",
    "   .XX..##OO..  ",
    "    .X.#OO..    ",
    "     ..O..      ",
    "      ..        ",
    "                " ]

$mini_page_xpm = [
    "16 16 4 1",
    "       c None s None",
    ".      c black",
    "X      c white",
    "o      c #808080",
    "                ",
    "   .......      ",
    "   .XXXXX..     ",
    "   .XoooX.X.    ",
    "   .XXXXX....   ",
    "   .XooooXoo.o  ",
    "   .XXXXXXXX.o  ",
    "   .XooooooX.o  ",
    "   .XXXXXXXX.o  ",
    "   .XooooooX.o  ",
    "   .XXXXXXXX.o  ",
    "   .XooooooX.o  ",
    "   .XXXXXXXX.o  ",
    "   ..........o  ",
    "    oooooooooo  ",
    "                " ]

# define xpms for toolbar
$toolbar_sample_xpm = [
    "20 19 5 1",
    ". c None",
    "# c #000000",
    "i c #ffffff",
    "s c #7f7f00",
```

```
  "y c #ffff00",
  ".....................",
  ".....................",
  ".....................",
  "..........###......",
  "..........#...#.#...",
  "..............##...",
  "...###........###...",
  "..#yiy#######.......",
  "..#iyiyiyiyi#.......",
  "..#yiyiyiyiy#.......",
  "..#iyiy###########..",
  "..#yiy#sssssssss#...",
  "..#iy#sssssssss#....",
  "..#y#sssssssss#.....",
  "..##sssssssss#......",
  "..###########.......",
  ".....................",
  ".....................",
  "....................." ]

# create Pixmap object from xpm data
def new_pixmap(window, background)
  pixmap, mask = Gdk::Pixmap::create_from_xpm_d(window, background, $toolbar_sample_xpm)
  wpixmap = Gtk::Pixmap::new(pixmap, mask)
end

# setup colors and fonts
$black   = Gdk::Color.new(0x0000, 0x0000, 0x0000)
$white   = Gdk::Color.new(0xFFFF, 0xFFFF, 0xFFFF)
$blue    = Gdk::Color.new(0x0000, 0x0000, 0xFFFF)
$green   = Gdk::Color.new(0x0000, 0xCCCC, 0x0000)
$magenta = Gdk::Color.new(0xFFFF, 0x0000, 0xFFFF)
$red     = Gdk::Color.new(0xFFFF, 0x0000, 0x0000)
$font = Gdk::Font.font_load("-adobe-courier-medium-r-normal--*-120-*-*-*-*-*-*")


# overload method for XML::DOM::NamedNodeMap
module XML
module DOM
class NamedNodeMap
  def to_h
    return {} if @nodes.nil?
    hash = {}
    @nodes.each do |k, v| hash[v.nodeName] = v.nodeValue end
    hash
  end
  def to_s
    return "" if @nodes.nil?
    str = ""
    @nodes.each do |k, v| str << v.to_s + ", " end
    str.chop.chop
  end
end
end
end

class MyWriter
  def initialize(tempfile)
    @tf = tempfile
  end
```

```
166
167   def visit_Document(document)
168     @tf.print "<?xml version=\"1.0\"?>\n"
169     @tf.print "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">\n"
170     document.children_accept(self)
171   end
172
173   def visit_Element(element)
174     attrs = ""
175     element.attributes.each do |attr|
176       attrs += " " + attr.to_s
177     end
178     @tf.print "<#{element.nodeName}#{attrs}>"
179     element.children_accept(self);
180     @tf.print "</#{element.nodeName}>"
181   end
182
183   def visit_Text(text)
184     @tf.print text.nodeValue
185   end
186
187   def visit_Comment(text) end
188
189 end
190
191 =begin
192 # overload method for NQXML::NamedAttributes
193 module NQXML
194   class NamedAttributes
195     def attributesToString
196       return "" if @attrs.nil?
197       str = ""
198       @attrs.each do |key, val| str << "#{key}=\"#{val.to_s}\", " end
199       return str.chop.chop # get rid of trailing ", "
200     end
201   end
202 end
203 =end
204
205
206 # split string (key-value list) into hash
207 class String
208   def to_h(delim=",")
209     tokens = self.split(delim)
210     hash = {}
211     tokens.each do |tok|
212       tok2 = tok.strip
213       ind = tok2.index("=")
214       key = tok2[0, ind]
215       val = tok2[ind+2..tok2.rindex("\"")-1]
216       hash[key] = val
217     end
218     hash
219   end
220 end
221
222
223 # simple dialog with label and OK button
224 class MyDialog < Gtk::Dialog
225   def initialize(title, label, button="OK")
226     super()
```

```
227     border_width(0)
228     set_usize(200, 100)
229     signal_connect("destroy") do destroy end
230     set_title(title)
231     set_modal(true)
232 
233     button = Gtk::Button::new(label)
234     button.flags |= Gtk::Widget::CAN_DEFAULT
235     button.signal_connect("clicked") do destroy end
236     action_area.pack_start(button)
237     button.grab_default
238 
239     label = Gtk::Label::new(label)
240     label.set_padding(10, 10)
241     vbox.pack_start(label)
242 
243     show_all
244   end
245 end
246 
247 
248 # somehow this is needed to destroy some widgets properly
249 module Destroyable
250   def destroy
251     super
252     @destroyed = true
253   end
254   def destroyed?
255     @destroyed
256   end
257 end
258 
259 
260 # show file selection dialog
261 class FileSelection < Gtk::FileSelection
262 
263   def initialize()
264     super("Choose a korpus index file")
265     signal_connect("destroy") do destroy end
266     set_modal(true)
267 
268     set_position(Gtk::WIN_POS_MOUSE)
269     hide_fileop_buttons
270 
271     ok_button.signal_connect("clicked") do
272       yield(get_filename)
273       destroy
274     end
275     cancel_button.signal_connect("clicked") do
276       destroy
277     end
278   end
279 
280 end
281 
282 
283 # notebook with customizable pages
284 class MyNotebook < Gtk::Notebook
285   include Destroyable
286 
287   # needs parent window for pixmaps
```

```
288   def initialize(window)
289     super()
290     @destroyed = false
291     signal_connect("destroy") do destroy end
292     signal_connect("switch_page") do |widget, page, num_page|
293       unless destroyed?
294         page_switch(widget, page, num_page)
295       end
296     end
297     border_width(5)
298     set_tab_pos(Gtk::POS_TOP)
299     @book_open, @book_open_mask =
300       Gdk::Pixmap::create_from_xpm_d(window, nil, $book_open_xpm)
301     @book_closed, @book_closed_mask =
302       Gdk::Pixmap::create_from_xpm_d(window, nil, $book_closed_xpm)
303   end
304 
305   private
306   def page_switch(widget, page, page_num)
307     oldpage = widget.cur_page
308     return if page == oldpage
309     page.tab_label.children[0].set(@book_open, @book_open_mask)
310     page.menu_label.children[0].set(@book_open, @book_open_mask)
311     if (oldpage)
312       oldpage.tab_label.children[0].set(@book_closed, @book_closed_mask)
313       oldpage.menu_label.children[0].set(@book_closed, @book_closed_mask)
314     end
315   end
316 
317   public
318   def create_page(buffer, widget)
319     child = Gtk::Frame::new(buffer)
320     child.border_width(5)
321 
322     vbox = Gtk::VBox::new(true, 0)
323     vbox.border_width(5)
324     child.add(vbox)
325 
326     hbox = Gtk::HBox::new(true, 0)
327     vbox.pack_start(hbox, false, true, 5)
328 
329     hbox.pack_start(widget)
330     child.show_all
331 
332     label_box = Gtk::HBox::new(false, 0)
333     pixwid = Gtk::Pixmap::new(@book_closed, @book_closed_mask)
334     label_box.pack_start(pixwid, false, true, 0)
335     pixwid.set_padding(3, 1)
336     label = Gtk::Label::new(buffer)
337     label_box.pack_start(label, false, true, 0)
338     label_box.show_all
339 
340     menu_box = Gtk::HBox::new(false, 0)
341     pixwid = Gtk::Pixmap::new(@book_closed, @book_closed_mask)
342     menu_box.pack_start(pixwid, false, true, 0)
343     pixwid.set_padding(3, 1)
344     label = Gtk::Label::new(buffer)
345     menu_box.pack_start(label, false, true, 0)
346     menu_box.show_all
347 
348     append_page_menu(child, label_box, menu_box)
```

```
  end

end


# show contents of file in scrollable window
class TextWindow < Gtk::Window
  def initialize(filename, colors=false)
    super()
    set_title("KBrowser - " + File.basename(filename))
    set_usize(500, 500)
    set_policy(true, true, false)
    signal_connect("destroy") do destroy end

    # create surrounding boxes
    box1 = Gtk::VBox.new(false, 0)
    add(box1)
    box1.show

    box2 = Gtk::VBox.new(false, 10)
    box2.border_width(10)
    box1.pack_start(box2, true, true, 0)
    box2.show

    scrolled_window = Gtk::ScrolledWindow.new(nil, nil)
    box2.pack_start(scrolled_window, true, true, 0)
    scrolled_window.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS)
    scrolled_window.show

    # create Text widget
    text = Gtk::Text.new(nil, nil)
    text.set_editable(false)
    scrolled_window.add(text)
    text.grab_focus
    text.show
    text.freeze

    # load content of give file into text widget
    begin
      buffer = nil
      File.open(filename, "r") do |infile|
        buffer = infile.read
      end
      if colors
        # split each line at color tags and segment/sentence boundaries
        buffer.each_line do |l|
          tok = l.split('\t')
          if tok.size == 3
            text.insert($font, $black, $white, tok[0]+"\t")
            text.insert($font, $blue, $white, tok[1]+"\t")
            if tok[2] =~ /^</
              text.insert($font, $red, $white, tok[2])
            elsif tok[2] =~ /^@/
              text.insert($font, $magenta, $white, tok[2])
            else
              text.insert($font, $black, $white, tok[2])
            end
          elsif l =~ /^</
            text.insert($font, $green, $white, l)
          else
            text.insert($font, nil, nil, l)
```

```
          end
        end
      else
        text.insert($font, nil, nil, buffer)
      end
      text.thaw
    rescue
      STDERR.puts "Hoppla: #{$!}"
    end
  end

end

# show contents of two files in scrollable windows
class DoubleWindow < Gtk::Window
  def initialize(filename)
    super()
    set_title("KBrowser - " + File.basename(filename))
    #set_usize(600, 500)
    set_policy(true, true, false)
    signal_connect("destroy") do destroy end
    signal_connect("delete_event") do false end
    @filename = [2]
    @filename[0] = filename
    if filename =~ /de/
      @filename[1] = filename.sub(/de/, "en")
    else
      @filename[1] = filename.sub(/en/, "de")
    end
    init
  end

  private
  def init
    # create surrounding boxes
    box1 = Gtk::VBox.new(false, 0)
    add(box1)
    box1.show

    #box2 = Gtk::HBox.new(false, 10)
    box2 = Gtk::HPaned.new()
    box2.border_width(10)
    box1.pack_start(box2, true, true, 0)
    box2.show

    # create Text widget
    text = [2]
    0.upto(1) do |i|
      scrolled_window = Gtk::ScrolledWindow.new(nil, nil)
      #box2.pack_start(scrolled_window, true, true, 0)
      if i == 0
        box2.add1(scrolled_window)
      else
        box2.add2(scrolled_window)
      end
      # Text widgets currently do not support horizontal scrollbars
      scrolled_window.set_policy(Gtk::POLICY_AUTOMATIC, Gtk::POLICY_ALWAYS)
=begin
      scrolled_window.signal_connect("motion_notify_event") do |w, c|
        print "x=#{c.x}, y=#{c.y}, x_root=#{c.x_root}, y_root=#{c.y_root}\n"
      end
```

```
=end
      scrolled_window.show

      text[i] = Gtk::Text.new(nil, nil)
      text[i].set_editable(false)
      text[i].set_usize(300, 400)
      scrolled_window.add(text[i])
      text[i].grab_focus
      text[i].show
      text[i].freeze
      read_file(text[i], i)
      text[i].thaw
    end
  end

  private
  def read_file(widget, nr)
    # load content of give file into text widget
    begin
      buffer = nil
      File.open(@filename[nr], "r") do |infile|
        buffer = infile.read
      end
      buffer.each_line do |l|
        tok = l.split('\t')
        if tok.size == 3
          widget.insert($font, nil, nil, tok[0]+" ")
        elsif l =~ /<SATZ>/
          widget.insert($font, nil, nil, "\n")
        elsif l =~ /<segmentgrenze>/
          widget.insert($font, nil, nil, "\n\n")
        end
      end
    rescue
      STDERR.puts "Hoppla: #{$!}"
    end
  end

end


# shows XML tree in a Gtk::CTree
class MyTree < Gtk::CTree

  attr_accessor :verbose

  def initialize(cols, n, window)
    super(cols, n)

    set_column_auto_resize(0, true);
    set_column_width(1, 200);
    set_selection_mode(Gtk::SELECTION_SINGLE);
    @line_style = Gtk::CTree::LINES_DOTTED;
    @node_style = Gtk::Style::new()
    @node_style.set_base(Gtk::STATE_NORMAL, 0, 45000, 55000)

    signal_connect("click_column") do |w,c| click_column(c) end

    @pixmap1, @mask1 = Gdk::Pixmap::create_from_xpm_d(window, style.white, $book_closed_xpm)
    @pixmap2, @mask2 = Gdk::Pixmap::create_from_xpm_d(window, style.white, $book_open_xpm)
    @pixmap3, @mask3 = Gdk::Pixmap::create_from_xpm_d(window, style.white, $mini_page_xpm)
```

```
532
533       set_usize(0, 300)
534       @filelist = [] # keeps all filenames collected from xml file
535       @verbose = false
536     end
537
538     # rebuild tree if data has changed
539     public
540     def rebuild_tree(filename)
541       @filename = filename
542       freeze
543       clear
544       @filelist.clear
545       begin
546         STDERR.print("Parsing XML file... (#{Time.now})") if @verbose
547         parser = XML::SimpleTreeBuilder.new(0, "ISO-8859-1")
548         doc =  parser.parse(File.new(@filename))
549         STDERR.puts(" ...done (#{Time.now})") if @verbose
550       rescue
551         STDERR.puts "Hoppla: #{$!}"
552         thaw
553         return
554       end
555       @root = doc.documentElement
556       STDERR.puts("Inserting parent") if @verbose
557       parent = insert_node(nil, nil, [@root.nodeName, @root.attributes.to_s], 5,
558                @pixmap1, @mask1, @pixmap2, @mask2, false, true)
559
560       node_set_row_data(parent, @node_style)
561       STDERR.print("Starting recursive build... (#{Time.now})") if @verbose
562       build_recursive(@root, parent)
563       STDERR.puts(" ...done (#{Time.now})") if @verbose
564       thaw
565       parser.done # clear up remaining garbage
566       @filelist # return filelist, so that CList can display it
567     end
568
569     # recursively walk through document model and fill tree/list with data
570     private
571     def build_recursive(node, parent)
572       sibling = nil
573       node.childNodes.each do |child|
574         case child.nodeType
575           when XML::DOM::Node::TEXT_NODE
576           str = child.nodeValue.strip
577           if str.length > 0
578             if child.parentNode.nodeName == "filename"
579               @filelist << [str, child.parentNode.attributes.to_h["format"]]
580             end
581             sibling = insert_node(parent, sibling,
582               ["", str],
583               5, @pixmap3, @mask3, nil, nil, true, false)
584           end
585         when XML::DOM::Node::COMMENT_NODE
586         else
587           sibling = insert_node(parent, sibling,
588             [child.nodeName, child.attributes.to_s],
589             5, @pixmap1, @mask1, @pixmap2, @mask2, false, false)
590           node_set_row_data(sibling, @node_style)
591           build_recursive(child, sibling)
592         end
```

```
593       end
594     end
595
596   =begin
597     def old_build_recursive(node, parent)
598       # old version for NQXML
599       sibling = nil
600       entity = node.entity
601       node.children.each do |child|
602         entity = child.entity
603         if entity.class == NQXML::Text
604           str = entity.to_s.strip
605           if str.length > 0
606             if child.parent.entity.name == "filename"
607               @filelist << [entity.to_s, child.parent.entity.attributesToString.to_h["format"]]
608             end
609             sibling = insert_node(parent, sibling,
610               ["", entity.to_s],
611               5, @pixmap3, @mask3, nil, nil, true, false)
612           end
613         elsif entity.class == NQXML::Tag
614           sibling = insert_node(parent, sibling,
615             [child.entity.name, child.entity.attributesToString],
616             5, @pixmap1, @mask1, @pixmap2, @mask2, false, false)
617           node_set_row_data(sibling, @node_style)
618           build_recursive(child, sibling)
619         end
620       end
621     end
622   =end
623
624     public
625     def subtree(id)
626       tf = nil
627       @root.childNodes.each do |child|
628         if child.nodeName == "document" && child.attributes.to_h["id"] == id
629           temproot = XML::DOM::Element.new("mapping", nil, child.clone)
630           tempdoc = XML::DOM::Document.new(temproot)
631           tf = Tempfile.new("preproc", File.dirname(@filename))
632           #STDERR.puts "Generating #{tf.path}"
633           tempdoc.accept(MyWriter.new(tf))
634           tf.close
635           break
636         end
637       end
638       tf
639     end
640
641     # user clicked on column title in tree page
642     private
643     def click_column(column)
644       if column == @ctree.sort_column then
645         if sort_type == Gtk::SORT_ASCENDING then
646           sort_type = Gtk::SORT_DESCENDING
647         else
648           sort_type = Gtk::SORT_ASCENDING
649         end
650       else
651         sort_column = column
652       end
653       sort_recursive(nil)
```

```
654   end
655
656 end
657
658
659 # shows filenames in a Gtk::CList
660 class MyList < Gtk::CList
661
662   def initialize(cols)
663     super(cols)
664     set_selection_mode(Gtk::SELECTION_SINGLE);
665     column_titles_passive
666     column_titles_show
667     set_column_auto_resize(0, true)
668     set_column_resizeable(1, false)
669   end
670
671   # rebuild list if data has changed
672   public
673   def rebuild_list(filelist)
674     if filelist.size > 0
675       freeze
676       clear
677       filelist.each do |v| append([v[0],v[1]]) end
678       thaw
679     end
680   end
681
682 end
683
684 # shows radio buttons to select options for preproc.py
685 class CheckBox < Gtk::Window
686   def initialize(title)
687     super(Gtk::WINDOW_TOPLEVEL)
688     set_title(title)
689     @choice = ["-i kbrowser", "-c"]
690     signal_connect("destroy") do destroy end
691     signal_connect("delete_event") do false end
692     init
693   end
694
695   private
696   def init
697     box1 = Gtk::VBox::new(false, 0)
698     box1.border_width(5)
699     add(box1)
700     box1.show
701
702     label = Gtk::Label::new("What do you want preproc to do?")
703     box1.pack_start(label, true, true, 0)
704     label.show
705
706     frame = Gtk::Frame::new("Task")
707     frame.border_width(5)
708     box1.pack_start(frame, true, true, 0)
709     frame.show
710
711     box2 = Gtk::VBox::new(false, 10)
712     box2.border_width(10)
713     frame.add(box2)
714     box2.show
```

```
715
716       button = Gtk::RadioButton::new(nil, "Convert")
717       button.signal_connect("toggled") do @choice = ["-i kbrowser", "-c"] end
718       box2.pack_start(button, true, true, 0)
719       button.show
720
721       button = Gtk::RadioButton::new(button.group, "Tag")
722        button.signal_connect("toggled") do @choice = ["-i kbrowser", "-n", "-a"] end
723       box2.pack_start(button, true, true, 0)
724       button.show
725
726       button = Gtk::RadioButton::new(button.group, "Align")
727       button.signal_connect("toggled") do @choice = ["-i kbrowser", "-n"] end
728       box2.pack_start(button, true, true, 0)
729       button.show
730
731       button = Gtk::RadioButton::new(button.group, "Enter DB")
732       button.signal_connect("toggled") do @choice = ["-i kbrowser"] end
733       box2.pack_start(button, true, true, 0)
734       button.show
735
736       separator = Gtk::HSeparator::new()
737       box1.pack_start(separator, false, true, 0)
738       separator.show
739
740       box2 = Gtk::HBox::new(false, 10)
741       box2.border_width(10)
742       box1.pack_start(box2, false, true, 0)
743       box2.show
744
745       button = Gtk::Button::new("Cancel")
746       button.signal_connect("clicked") do destroy end
747       box2.pack_start(button, true, true, 0)
748       button.show
749
750       button = Gtk::Button::new("Run")
751       button.signal_connect("clicked") do
752         yield(@choice)
753         destroy
754       end
755       box2.pack_start(button, true, true, 0)
756       button.set_flags(Gtk::Widget::CAN_DEFAULT)
757       button.grab_default
758       button.show
759
760     end
761   end
762
763   # the main window which does all the work
764   class MainWindow < Gtk::Window
765
766     def initialize(title, verbose=false)
767       super(Gtk::WINDOW_TOPLEVEL)
768       @filename = nil
769       @title = title
770       @verbose = verbose
771       set_title(@title)
772       set_policy(true, true, false);
773       set_name("main window")
774       set_usize(600, 400)
775       set_uposition(40, 40)
```

```
776
777       signal_connect("destroy") do Gtk.main_quit end
778       signal_connect("delete_event") do false end
779
780       init
781     end
782
783     # set filename, triggers rebuilding of tree/list
784     def set_filename(fnm)
785       @filename = fnm
786       @dirname = File.dirname(fnm)
787       set_title(@title + " - " + fnm)
788       filelist = @ctree.rebuild_tree(fnm)
789       @clist.rebuild_list(filelist)
790       message("Loaded index file " + @filename)
791     end
792
793     # initialize GUI
794     private
795     def init
796       @box = Gtk::VBox::new(false, 0)
797       add(@box)
798
799       create_menu
800       create_toolbar
801       create_notebook
802       create_statusbar
803
804       @box.show
805     end
806
807     # create a statusbar for useless messages
808     private
809     def create_statusbar
810       @statusbar = Gtk::Statusbar::new()
811       @box.pack_end(@statusbar, false, false, 0)
812       @main_id = @statusbar.get_context_id("Main")
813       #@toolbar_id = @statusbar.get_context_id("Toolbar")
814       @statusbar.push(@main_id, "Welcome to KBrowser!")
815       @statusbar.show
816     end
817
818     # create a menu bar
819     private
820     def create_menu
821       menubar = Gtk::MenuBar::new()
822       menu = Gtk::Menu::new()
823
824       # create File menu
825       menuitem = Gtk::MenuItem::new("File")
826       submenu = Gtk::Menu::new()
827       smenuitem = Gtk::MenuItem::new("Open")
828       submenu.append(smenuitem)
829       smenuitem.signal_connect("activate") do
830         f = FileSelection.new {|f| set_filename(f)}
831         f.show
832       end
833       smenuitem = Gtk::MenuItem::new("Quit")
834       smenuitem.signal_connect("activate") do Gtk.main_quit end
835       submenu.append(smenuitem)
836
```

```
837       # connect to menubar
838       menuitem.set_submenu(submenu)
839       menubar.append(menuitem)
840
841       # create Help menu
842       menuitem = Gtk::MenuItem::new("Help")
843       submenu = Gtk::Menu::new()
844       smenuitem = Gtk::MenuItem::new("About")
845       smenuitem.signal_connect("activate") do
846         MyDialog.new("About KBrowser", "KBrowser rulez!")
847       end
848       submenu.append(smenuitem)
849
850       # connect to menubar
851       menuitem.set_submenu(submenu)
852       menuitem.right_justify
853       menubar.append(menuitem)
854
855       menubar.show_all
856       @box.pack_start(menubar, false, true, 0)
857     end
858
859     # create a toolbar with nice buttons
860     private
861     def create_toolbar
862       tbox = Gtk::HBox::new(false, 5)
863       @box.pack_start(tbox, false, false, 0)
864       tbox.show
865       toolbar = Gtk::Toolbar::new(Gtk::ORIENTATION_HORIZONTAL, Gtk::TOOLBAR_BOTH)
866       toolbar.set_button_relief(Gtk::RELIEF_NONE)
867       toolbar.set_space_style(Gtk::Toolbar::SPACE_LINE)
868       realize
869       toolbar.append_item("Original", "View original file in korpus",
870                   "Toolbar/Original",
871                   new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
872                   nil) do
873         show_file(5, false)
874       end
875       toolbar.append_item("Ascii", "View ASCII file in korpus",
876                   "Toolbar/Ascii",
877                   new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
878                   nil) do |w|
879         show_file(1, false)
880       end
881       toolbar.append_item("Tagged", "View tagged file in korpus",
882                   "Toolbar/Tagged",
883                   new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
884                   nil) do
885         show_file(2, true)
886       end
887       toolbar.append_item("Aligned", "View aligned file in korpus",
888                   "Toolbar/Aligned",
889                   new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
890                   nil) do
891         show_file(3, true)
892       end
893       toolbar.append_item("Both", "View both aligned files in korpus",
894                   "Toolbar/Both",
895                   new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
896                   nil) do
897         show_file(4, true)
```

```
898       end
899       toolbar.append_space
900       toolbar.append_item("Preproc", "Run preproc.py on selected document",
901                   "Toolbar/Preproc",
902                   new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
903                   nil) do
904         preproc
905       end
906       tbox.add(toolbar)
907       toolbar.show
908     end
909
910     # construct local subtree
911     private
912     def preproc
913       return if @notebook.get_current_page == 1
914       @ctree.each_selection do |s|
915         if @ctree.node_get_pixtext(s, 0)[0] == "document"
916           id = @ctree.node_get_text(s, 1).to_h["id"]
917           next unless id
918           tf = @ctree.subtree(id)
919           cb = CheckBox.new("Select task") do |c|
920             @statusbar.push(@main_id, "Running preproc.py")
921             args = c << tf.path
922             launch_external($preproc_cmd, args)
923           end
924           cb.show
925         end
926       end
927     end
928
929     # get filename of selected entry from tree/list
930     private
931     def get_selected
932       fnm = ""
933       format = ""
934       case @notebook.get_current_page
935         when 0
936           @ctree.each_selection do |s|
937             if s.leaf? && @ctree.node_get_pixtext(s.parent, 0)[0] == "filename"
938               # only works if filename selected
939               fnm = @ctree.node_get_text(s, 1)
940               fnm = yield(fnm) if block_given? # get correct files suffix
941               format = @ctree.node_get_text(s.parent, 1).to_h["format"]
942             end
943           end
944         when 1
945           @clist.each_selection do |s|
946             fnm = @clist.get_text(s, 0)
947             fnm = yield(fnm) if block_given?
948             format = @clist.get_text(s, 1)
949           end
950       end
951       if fnm.length > 0
952         return [fnm, format]
953       else
954         return nil
955       end
956     end
957
958     # show original/ascii/tagged/aligned korpus file
```

```
959   private
960   def show_file(type, color=false)
961     case type
962       when 1
963         fnm, format = get_selected {|f| f[0, f.rindex(".") + 1] + "asc2"}
964       when 2
965         fnm, format = get_selected {|f| f[0, f.rindex(".") + 1] + "tag"}
966       when 3,4
967         fnm, format = get_selected {|f| f[0, f.rindex(".") + 1] + "tag.al"}
968       when 5
969         fnm, format = get_selected
970     end
971     return unless fnm
972     cfnm = @dirname + File::SEPARATOR + fnm
973     if FileTest.exist?(cfnm)
974       succesful = true
975       if type == 5
976         # launch external viewer
977         if format == "plain" || format == "tagged" || format == "sgml"
978           TextWindow.new(cfnm).show
979         elsif format == "ps"
980           launch_external($ps_viewer, cfnm)
981         elsif format == "pdf"
982           launch_external($pdf_viewer, cfnm)
983         elsif format == "html"
984           launch_external($html_viewer, cfnm)
985         elsif format == "unknown" || format == "doc"
986           succesful = false
987         end
988       elsif type == 4
989         DoubleWindow.new(cfnm).show
990       else
991         # open text window
992         TextWindow.new(cfnm, color).show
993       end
994       if succesful
995         message("Launched viewer for " + fnm)
996       else
997         message("Don't know what to do with file " + fnm)
998       end
999     else
1000      message("File not found: " + cfnm)
1001    end
1002  end
1003
1004  # launch external viewer and trap signal of its death
1005  private
1006  def launch_external(cmd, args)
1007    trap("CLD") do
1008      pid = Process.wait
1009      message("Child pid #{pid}: terminated")
1010    end
1011    exec(cmd, *args) if fork == nil
1012  end
1013
1014  private
1015  def create_notebook
1016    @notebook = MyNotebook.new(window)
1017    @box.pack_start(@notebook, true, true, 0)
1018    @notebook.create_page("Tree View", create_tree)
1019    @notebook.create_page("List View", create_list)
```

```
    @notebook.show
  end

  private
  def create_list
    vbox = Gtk::VBox::new()
    scrolled_win = Gtk::ScrolledWindow::new()
    scrolled_win.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS)
    vbox.pack_start(scrolled_win)

    @clist = MyList.new(["File", "Format"])
    @clist.signal_connect("select_row") do message("") end
    scrolled_win.add(@clist)

    vbox
  end

  private
  def create_tree
    vbox = Gtk::VBox::new()
    scrolled_win = Gtk::ScrolledWindow::new()
    scrolled_win.border_width(5)
    scrolled_win.set_policy(Gtk::POLICY_AUTOMATIC, Gtk::POLICY_ALWAYS)
    vbox.pack_start(scrolled_win)

    @ctree = MyTree.new(["Tree", "Info"], 0, window)
    @ctree.verbose = @verbose
    @ctree.signal_connect_after("button_press_event") do message("") end
    scrolled_win.add(@ctree)

    hbox = Gtk::HBox::new(false, 5)
    vbox.pack_start(hbox, false, false, 0)

    tooltips = Gtk::Tooltips::new()

    button = Gtk::Button::new("Expand All")
    hbox.pack_start(button)
    button.signal_connect("clicked") do
      @ctree.expand_recursive(nil)
      message("Expanded XML Tree")
    end
    tooltips.set_tip(button, "Expand all nodes of the tree", "Buttons/Expand")

    button = Gtk::Button::new("Collapse All")
    hbox.pack_start(button)
    button.signal_connect("clicked") do
      @ctree.collapse_recursive(nil)
      message("Collapsed XML Tree")
    end
    tooltips.set_tip(button, "Collapse all nodes of the tree", "Buttons/Collapse")

    vbox
  end

  # set new message in status bar
  def message(text)
    @statusbar.pop(@main_id) unless @statusbar.messages.empty?
    @statusbar.push(@main_id, text)
  end

end
```

```
1081
1082
1083 def main
1084   window = MainWindow.new("KBrowser", false)
1085   window.show()
1086   if ARGV.length == 1
1087     window.set_filename(ARGV[0])
1088   end
1089   Gtk::main()
1090 end
1091
1092 main
```

I.3 XML Tools

I.3.1 mapping.dtd

DTD für `index.xml` (s. C.2.1)

```
1  <?xml encoding="ISO-8859-1"?>
2
3  <!--
4          DTD for file name mapping of German and English corpus documents
5          KOKS project
6          Last change: Arno Erpenbeck, 20.07.2001
7  -->
8
9  <!-- $Header: /opt/cvs/share/mapping.dtd,v 1.10 2001/10/25 11:51:18 koks-cvs Exp $ -->
10
11 <!-- some information remains unknown -->
12 <!ENTITY % unknown.att "unknown">
13
14 <!-- flag indicating whether a document can be user for further processing -->
15 <!ENTITY % use.att "yes|no">
16
17 <!-- document kind is either lexicon, phrase, or text -->
18 <!ENTITY % kind.att "lexicon|phrase|text">
19
20 <!-- documents can be either German (de) or English (en)
21      or both languages in one file (bi) -->
22 <!ENTITY % lang.att "de|en|bi">
23
24 <!-- each part can use its own tag set -->
25 <!ENTITY % set.att "ims|lql">
26
27 <!-- allowed file formats, list should be extended as needed -->
28 <!ENTITY % format.att "plain|html|sgml|pdf|ps|doc|tagged|%unknown.att;">
29
30 <!-- recognized encodings for text files -->
31 <!ENTITY % enc.att "iso-8859-1|%unknown.att;">
32
33 <!-- allowed types for document source -->
34 <!ENTITY % type.att "url|cdrom|%unknown.att;">
35
36 <!-- allowed methods for document checksums -->
37 <!ENTITY % method.att "cksum|md5|sha1">
38
39 <!ELEMENT mapping (document+)>
40
41 <!-- each document consists of one or more parts and additional info -->
42 <!ELEMENT document (part+,info?)>
```

```
<!ATTLIST document
                  id ID #IMPLIED
                  use (%use.att;) #IMPLIED
                  kind (%kind.att;) #IMPLIED>

<!-- each part consists of one or more files in a specified language -->
<!ELEMENT part (file+)>
<!ATTLIST part
               lang (%lang.att;) #REQUIRED
               tagset (%set.att;) #IMPLIED>

<!-- a file is identified by its name and an optional checksum -->
<!ELEMENT file (filename,checksum?)>

<!-- filename contains the relative path to the described file,
     also there is an optional encoding tag  -->
<!ELEMENT filename (#PCDATA)>
<!ATTLIST filename
                  format (%format.att;) #REQUIRED
                  enc (%enc.att;) #IMPLIED>

<!-- checksum can be used to uniquely identify documents -->
<!ELEMENT checksum (#PCDATA)>
<!ATTLIST checksum method (%method.att;) #REQUIRED>

<!-- the info tag can be further refined, e.g. for source information -->
<!ELEMENT info (author,comment,date,source)>

<!ELEMENT author (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT date (#PCDATA)>

<!-- type of source can remain unspecified -->
<!ELEMENT source (#PCDATA)>
<!ATTLIST source type (%type.att;) #IMPLIED>

```

I.3.2 meta.dtd

DTD für `metaindex.xml` (s. C.2.1)

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
        DTD for meta index listing all index.xml files in corpus directory
        KOKS project
        Last change: Arno Erpenbeck, 19.07.2001
-->

<!-- $Header: -->

<!ELEMENT meta-index (korpus+)>

<!ELEMENT korpus (file*)>
<!ATTLIST korpus
                 name CDATA #REQUIRED>

<!ELEMENT file (#PCDATA)>
```

I.3.3 idx-ls.tcl

Listet Inhalt von `index.xml` auf (s. C.1.3)

```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh8.3 "$0" "$@"

# $Header: /opt/cvs/bin/idx-ls.tcl,v 1.1 2001/10/21 15:20:24 koks Exp $

# list contents of an index.xml file
# KOKS project
# Arno Erpenbeck, 03.04.2001

# make sure koksUtil can be found
lappend auto_path /home/koks/share/tcl

# load packages
package require xml
package require koksUtil

# set up global variables
set version "0.4 - 03.04.2001"
set fnm "index.xml"
set count(total) 0
set count(part) 0
set flag(filter) ""
set flag(form) 0
set flag(lang) 0
set flag(file) 0
set flag(skip) 0
set flag(stat) 0
set flag(terse) 0
set flag(changed) 0
set flag(checksum) 0
set line(checkfile) ""
set line(content) ""

# --------------------------------------------------------------------------

# handle opening tags
proc HandleStart {name attlist} {
    global flag line count
    switch -exact -- $name {
        document {
            # only print newline if more than one document
            if $count(part) {puts ""}
            incr count(total)
        }
        part {
            set l [lindex $attlist 1]
            if $flag(lang) {
                # only print parts of matching language
                if {![string compare $l $flag(filter)]} {
                    # current lang matches specified
                                        if !$flag(terse) {
                                                set line(content) "$l: "
                                        }
                } else {
                    # skip this part
                    set flag(skip) 1
                }
            } elseif !$flag(terse) {
                                set line(content) "$l: "
            } else {
```

```
                                set line(content) ""
            }
        }
        checksum {
                        # if "-c" flag, test for checksum (currently only md5)
            if {$flag(changed) && ![string compare [lindex $attlist 1] "md5"]} {
                                set flag(checksum) 1
                        }
        }
        filename {
            if !$flag(skip) {
                # only print if part not skipped
                if $flag(form) {
                                        append line(content) "([lindex $attlist 1]) "
                                }
                set flag(file) 1
            }
        }
    }
}

# handle closing tags
proc HandleEnd {name} {
    global flag line count
    switch -exact -- $name {
        mapping {
                        if $flag(stat) {
                                puts "--> $count(total) documents ($count(part) parts)"
                        }
                }
        part {
            if !$flag(skip) {
                                puts $line(content)
                    incr count(part)
                        }
            set flag(skip) 0
        }
        checksum {
                        # clear flags
                set flag(checksum) 0
                        set line(checkfile) ""
                }
        filename {
                        set flag(file) 0
                }
    }
}

# handle text data between tags
proc HandleText {data} {
    global flag line
    if $flag(file) {
        # we are inside a <filename> tag
                #set line(checkfile) [string trim $data]
                if !$flag(changed) {
                        # print out every filename
                append line(content) [string trim $data]
                } else {
                        # keep filename, only print if changed
                        set line(checkfile) [string trim $data]
                }
```

```
    } elseif $flag(checksum) {
        # we are inside a <checksum> tag
        # compute md5 checksum and complain if changed
        set cfnm $line(checkfile)
        if {[catch {set result [lindex [split [exec md5 $line(checkfile)]] 3]} err]} {
            puts stderr "Error: cannot generate checksum using md5!"
        }
        if !$flag(changed) {
            append line(content) $line(checkfile)
        } elseif {![string compare [string trim $data] $result]} {
                        # checksum ok, skip entry
            set flag(skip) 1
        } else {
                        # checksum changed, print out filename
            append line(content) $line(checkfile)
        }
    }
}


# check if file obeys to correct DTD
proc HandleDocType {docelement publicID systemID internalDTD} {
    if {[string compare $systemID "/home/koks/share/mapping.dtd"]} {
        puts stderr "index file is of wrong DTD: $systemID"
        exit 2
    }
}

# print help message
proc Help {} {
    puts "Usage: idx-ls.tcl \[-c\] \[-f\] \[-s\] \[-t\] \[-l <lang>\] \[index-file\]"
    puts "-c: compute checksum and only list changed files"
        puts "-f: print out file format of each file"
        puts "-l <lang>: only list entries of language <lang> (de, en, bi)"
        puts "-s: print some statistics in the end"
        puts "-t: terse mode, only list file names"
    exit
}

# print version info
proc Version {} {
    global version
    puts $version
    exit
}

# ------------------------------------------------------------------------

# check command line args
foreach arg $argv {
    switch -exact -- $arg {
        -h {Help}
        -v {Version}
        -c {set flag(changed) 1}
        -f {set flag(form) 1}
        -l {set flag(lang) 1}
                -s {set flag(stat) 1}
                -t {set flag(terse) 1}
        de {if $flag(lang) {set flag(filter) $arg}}
        en {if $flag(lang) {set flag(filter) $arg}}
        bi {if $flag(lang) {set flag(filter) $arg}}
```

```
        default {set fnm $arg}
    }
}

# open index file
set in [koksUtil::openFile $fnm]

# create a new xml parser
set p [xml::parser]

# configure parser
$p configure -doctypecommand HandleDocType
$p configure -elementstartcommand HandleStart
$p configure -elementendcommand HandleEnd
$p configure -characterdatacommand HandleText

# parse document
$p parse [read $in]

# close file
koksUtil::closeFile $in
```

I.3.4 idx-add.tcl

Generiert `index.xml` (s. C.1.3)

```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh8.3 "$0" "$@"

# $Header: /opt/cvs/bin/idx-add.tcl,v 1.1 2001/10/21 15:20:24 koks Exp $

# create an index.xml file
# KOKS project
# Arno Erpenbeck, 16.05.2001

# set up global variables
set version "0.3 - 16.05.2001"
set valid(format) [list plain html sgml pdf ps doc unknown]
set valid(type) [list url cdrom unknown]
set valid(lang) [list de en bi]
set valid(use) [list yes no]
set count 0
set small 0
set files(1) [list]
set files(md5) 0
set lang(1) ""
set useFlag ""
set info(author) ""
set info(comment) ""
set info(date) ""
set info(source) ""
set info(type) ""

# ---------------------------------------------------------------------------

# print xml file header
proc Header {} {
  global useFlag
  puts "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>"
  puts "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">"
```

```
  puts "<mapping>"
  if [string length $useFlag] {
    puts "<document use=\"$useFlag\">"
  } else {
    puts "<document>"
  }
}

# print small xml file header
proc SmallHeader {} {
  global useFlag
  if [string length $useFlag] {
    puts "<document use=\"$useFlag\">"
  } else {
    puts "<document>"
  }
}

# print xml file footer
proc Footer {} {
  puts "</document>"
  puts "</mapping>"
}

# print small xml file footer
proc SmallFooter {} {
  puts "</document>"
}

# write the different parts
proc WriteParts {count} {
  global valid files lang
  for {set i 1} {$i <= $count} {incr i} {
    puts "<part lang=\"$lang($i)\">"
    foreach {name form} $files($i) {
      if {[Allowed $form $valid(format)]} {
        puts "<file>"
        puts "<filename format=\"$form\">$name</filename>"
        if $files(md5) {
          if {[catch {set check [lindex [split [exec md5 $name]] 3]} err]} {
            puts stderr "Error: cannot generate checksum using md5!"
          } else {
            puts "<checksum method=\"md5\">$check</checksum>"
          }
        }
        puts "</file>"
      } else {
        puts stderr "Warning: format \"$form\" not valid, entry skipped!"
      }
    }
    puts "</part>"
  }
}

# write the info part
proc WriteInfo {} {
  global valid info
  puts "<info>"
  puts "<author>$info(author)</author>"
  puts "<comment>$info(comment)</comment>"
  puts "<date>$info(date)</date>"
```

```
  if {[Allowed $info(type) $valid(type)]} {
    puts "<source type=\"$info(type)\">$info(source)</source>"
  } else {
    puts "<source>$info(source)</source>"
  }
  puts "</info>"
}

# check if given argument is valid
proc Allowed {format allowed} {
  foreach v $allowed {
    if {![string compare $v $format]} {
      return 1
    }
  }
  return 0
}

# print help message
proc Help {} {
  global valid
  puts "Usage:\nidx-add.tcl \[-x\] \[info-options\] \[-u <use>\] -p -l <lang> <file1> <format1> <file2> <form
  puts "General Options:"
  puts "-x:  short mode, skip document header and footer"
  puts "-p:  start a new part"
  puts "-m:  generate id (checksum) using md5"
  puts "-l:  set language of part"
  puts "-u:  mark document for use\n"
  puts "Info Options:"
  puts "-a:  set author"
  puts "-c:  set comment"
  puts "-d:  set date"
  puts "-s:  set source"
  puts "-t:  set source type\n"
  puts "<source_type> may be one of \"$valid(type)\""
  puts "<format> may be one of \"$valid(format)\""
  puts "<lang> may be one of \"$valid(lang)\""
  puts "<use> may be one of \"$valid(use)\""
  exit
}

# print version info
proc Version {} {
  global version
  puts $version
  exit
}

# ------------------------------------------------------------------------

# check if any arguments given
if {![llength $argv]} {Help}

# process command line arguments
set flag error
foreach arg $argv {
  switch -exact -- $arg {
    -a {set flag author}
    -c {set flag comment}
    -d {set flag date}
    -h {Help}
```

```
    -l {set flag lang}
    -m {set files(md5) 1}
    -p {incr count; set flag part}
    -x {set small 1}
    -s {set flag source}
    -t {set flag type}
    -u {set flag use}
    -v {Version}
    default {
      switch -exact -- $flag {
        author {set info(author) $arg}
        comment {set info(comment) $arg}
        date {set info(date) $arg}
        error {Help}
        lang {set lang($count) $arg; set flag part}
        part {lappend files($count) $arg}
        source {set info(source) $arg}
        type {set info(type) $arg}
        use {set useFlag $arg}
        default {Help}
      }
    }
  }
}

# generate xml file and print to stdout
if $small {SmallHeader} else {Header}
WriteParts $count
WriteInfo
if $small {SmallFooter} else {Footer}
```

I.3.5 idx-repair.tcl

Repariert Rumpf-Datei (s. C.1.3)

```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh8.3 "$0" "$@"

# $Header: /opt/cvs/bin/idx-repair.tcl,v 1.1 2001/10/21 15:20:24 koks Exp $

# repair an index.xml file
# KOKS project
# Arno Erpenbeck, 11.01.2001

# load packages
lappend auto_path /home/koks/share/tcl
package require koksUtil


if $argc {
  set in [koksUtil::openFile [lindex $argv 0]]
} else {
  set in [koksUtil::openFile "index.xml"]
}

puts "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>"
puts "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">"
puts "<mapping>"
puts [read $in]
puts "</mapping>"
```

```
27
28 koksUtil::closeFile $in
```

I.3.6 meta-ls.rb

Listet Meta-Index `metaindex.xml` (s. C.1.3)

```
1  #!/usr/local/bin/ruby
2
3  require "xmlparser"
4
5  # $Header: /opt/cvs/bin/meta-ls.rb,v 1.2 2001/07/23 14:23:46 koks-cvs Exp $
6
7  # list contents of an metaindex.xml file
8  # KOKS project
9  # Arno Erpenbeck, 23.07.2001
10
11
12 class MyXMLParser < XMLParser
13
14   def initialize(enc)
15     super(enc)
16     @inside = false
17   end
18
19   def character(data)
20     print "\t", data.strip, "\n" if @inside
21   end
22
23   def startElement(name, attrs)
24     if name == "korpus"
25       puts attrs["name"]
26     elsif name == "file"
27       @inside = true
28     end
29   end
30
31   def endElement(name)
32     @inside = false if name == "file"
33   end
34
35 end
36
37 $0 = File.basename($0)
38 if ARGV.length == 0
39   inf = "metaindex.xml"
40 elsif ARGV.length == 1
41   inf = ARGV[0]
42 else
43   STDERR.puts "Usage: #{$0} [meta-index]"
44   exit 1
45 end
46 begin
47   parser = MyXMLParser.new("ISO-8859-1")
48   f = File.open(inf, "r")
49   parser.parse(f.read)
50   f.close
51 rescue
52   STDERR.puts "Hoppla: #{$!}"
53   exit 1
54 end
```

I.4 Sprachklassifikation

I.4.1 ngram.rb

N-Gramme erstellen (s. C.1.4)

```
#!/usr/local/bin/ruby

# $Header: /opt/cvs/artus/ngram.rb,v 1.3 2001/03/28 09:10:52 koks-cvs Exp $

# ngram.rb - build up n-gram table for text file
#
# KOKS project
# Arno Erpenbeck, 01.03.2001

require 'optparse'
require 'math/statistics'

# count occurences of n-grams
class Hash
  include Math::Statistics
  Hash::default_block = lambda{|i,j| j}
  def inc(key)
    prev = fetch(key, 0)
    store(key, prev+1)
  end
end

# make n-grams of arbitrary size of a given string
class String

  def normalize
    self.gsub(/[äöüÄÖÜß]/,'#').gsub(/\s/,' ').downcase
  end

  def ngrams(len=1)
    ngrams = []
    (0..size - len).each do |n|
      ng = self[n...n+len]
      if ng =~ /[a-zäöüßA-ZÄÖÜ\s]{3,3}/
        ngrams.push(ng)
        yield ng if block_given?
      end
    end
    ngrams
  end

end

# read file and build up n-grams
class Shredder

  def initialize(infile, outfile, bytes=-1, normalize=false)
    @infile = infile
    @outfile = outfile
    @bytes = bytes
    @normalize = normalize
  end

  def run(len=3)
    count = 0
    hash = {}
```

```
    if @normalize
      while line = @infile.gets
        line.ngrams(len) do |s| hash.inc(s.normalize) end
        if @bytes > 0
          count += line.length
          break if count >= @bytes
        end
      end
    else
      while line = @infile.gets
        line.ngrams(len) do |s| hash.inc(s) end
        if @bytes > 0
          count += line.length
          break if count >= @bytes
        end
      end
    end
    s = Float.induced_from(hash.size)
    #hash.sort.each {|k,v| @outfile.print k,":",v/s,"\n"}
    hash.sort.each {|k,v| @outfile.print k,":",v,"\n"}
    #@outfile.print hash.avg,"\n"
  end

end

# check command line args
$0 = File.basename($0)
vars = {:len => 3, :bytes => -1, :norm => false}
parser = OptionParser.new do |q|
  q.banner = "Usage: #{$0} [options]\n"
  q.on_tail("-h", "--help", "show this message") {puts q; exit}
  q.on("-i", "--input=FILENAME", String,
    "input file name") {|vars[:infile]|}
  q.on("-o", "--output=FILENAME", String,
    "output file name") {|vars[:outfile]|}
  q.on("-b", "--bytes=VALUE", Integer,
    "read at most VALUE bytes") {|vars[:bytes]|}
  q.on("-l", "--len=VALUE", Integer,
    "build n-grams of VALUE length") {|vars[:len]|}
  q.on("-n", "--normalize",
    "normalize n-grams") {|vars[:norm]|}
end
begin
  parser.parse!(ARGV)
rescue OptionParser::ParseError
  STDERR.puts parser
  exit 1
end

begin
  # open input and output files
  if vars[:infile]
    infile = open(vars[:infile], "r")
  else
    infile = STDIN
  end
  if vars[:outfile]
    outfile = open(vars[:outfile], "w+")
  else
    outfile = STDOUT
  end
```

```
  Shredder.new(infile, outfile, vars[:bytes], vars[:norm]).run(vars[:len])
rescue
  STDERR.puts "Something bad happened: #{$!}\n"
ensure
  # close file if necessary
  infile.close if vars[:infile]
  outfile.close if vars[:outfile]
end

```

I.4.2 classifier.rb

Klassifiziert Daten (s. C.1.4)

```
# $Header: /opt/cvs/artus/classifier.rb,v 1.2 2001/03/12 14:13:34 koks-cvs Exp $

# classifier.rb - try to identify the language of a text
#
# KOKS project
# Arno Erpenbeck, 11.03.2001

# find index of smallest/biggest value in array
class Array
  def max(start = 0)
    idx = start
    max = self[idx]
    self.each_index do |i|
      if self[i] > max
        max = self[i]
        idx = i
      end
    end
    idx
  end

  def min(start = 0)
    idx = start
    min = self[idx]
    self.each_index do |i|
      if self[i] < min
        min = self[i]
        idx = i
      end
    end
    idx
  end

end

# read file and build up n-grams
class Classifier

  def initialize(datafile, verbose=false)
    @datafile = datafile
    @verbose = verbose
    @q = []
    @sum = []
    @id = []
  end

  # read training data files
```

```
48    def read_data
49      # read file containing ids and names of training data files
50      fl = []
51      begin
52        dfin = open(@datafile, "r")
53        dfin.read.scan(/([a-z]+)\s([\w.\/-]+)/) do |id, fnm|
54          fl << fnm
55          @id << id
56        end
57      rescue
58        STDERR.puts "Something bad happened: #{$!}\n"
59      rescue
60        dfin.close
61      end
62      # open each file containg n-grams of training data
63      fl.each_index do |idx|
64        print "Reading data file ’", fl[idx], "’..." if @verbose
65        @q[idx] = {}
66        @sum[idx] = 0
67        begin
68          infile = open(fl[idx], "r")
69          infile.read.scan(/([a-z# ]+):(\d+)/) do |k, v|
70            @q[idx].store(k, v)
71            @sum[idx] += v.to_i
72          end
73        rescue
74          STDERR.puts "Something bad happened: #{$!}\n"
75        ensure
76          infile.close if infile
77        end
78        print "(",@q[idx].size,", ",@sum[idx], ")\n" if @verbose
79      end
80    end
81
82    # smooth training data
83    def smooth_data
84      print "Smoothing training data...\n" if @verbose
85      @q.each_index do |idx1|
86        @q[idx1].each_key do |k|
87          @q.each_index do |idx2|
88            if idx1 != idx2 and !@q[idx2].has_key?(k)
89              @q[idx2].store(k, 0.5)
90              @sum[idx2] += 0.5
91            end
92          end
93        end
94      end
95      print "Normalizing training data...\n" if @verbose
96      @q.each_index do |idx|
97        div = 1.0 / @sum[idx]
98        @q[idx].each do |k, v|
99          @q[idx].store(k, v.to_f * div)
100       end
101     end
102   end
103
104   # classify test data from hash
105   def classify(p)
106
107     # clean test data
108     print "Cleaning test data...\n" if @verbose
```

```
    p.each_key do |k|
      delete = TRUE
      @q.each_index do |i|
        if @q[i].has_key?(k)
          delete = FALSE
          break
        end
      end
      if delete
        p.delete(k)
      end
    end

    # normalize data
    print "Normalizing test data...\n" if @verbose
    sum = 0
    p.each do |k,v|
      sum += v.to_i
    end
    div = 1.0 / sum
    p.each do |k,v|
      p.store(k, v.to_f * div)
    end

    # calculate Kullback-Leibler distance
    print "Calculating distances...\n" if @verbose
    dist = Array.new(@q.size)
    @q.each_index do |i|
      dist[i] = 0.0
      p.each do |k, v|
        w = v.to_f
        #print "Ohje: ",k,"\n" if !@q[i].has_key?(k)
        z = @q[i].fetch(k).to_f
        dist[i] = dist[i] + (w * Math.log(w/z))
      end
      print "dist[",i,"] = ",dist[i],"\n" if @verbose
    end

    # find minimum -> return language classification
    @id[dist.min(1)]

  end

end
```

I.4.3 cfile.rb

Sprache einer Textdatei ermittlen (s. C.1.4)

```
#!/usr/local/bin/ruby

# $Header: /opt/cvs/artus/cfile.rb,v 1.2 2001/03/28 09:10:52 koks-cvs Exp $

# cfile.rb - try to identify the language of a text (file version)
#
# KOKS project
# Arno Erpenbeck, 11.03.2001

require 'optparse'
require 'classifier'

```

```
class FileReader

  def initialize(testfile, classifier, verbose=false)
    @testfile =  testfile
    @classifier = classifier
    @verbose = verbose
  end

  # read test data files and classify each file
  def classify_input
    fl = []
    # read file containing names of test data files
    begin
      dfin = open(@testfile, "r")
      while dfin.gets
        fl << $_.chomp!
      end
    rescue
      STDERR.puts "Something bad happened: #{$!}\n"
    rescue
      dfin.close
    end

    p = {}
    fl.each do |fnm|
      # open each test data file and classify
      print "Reading test file '", fnm, "'...\n" if @verbose
      p.clear
      begin
        infile = open(fnm, "r")
        infile.read.scan(/([a-z# ]+):(\d+)/) {|k, v| p.store(k, v)}
      rescue
        STDERR.puts "Something bad happened: #{$!}\n"
      ensure
        infile.close if infile
      end
      # call classifier
      id = @classifier.classify(p)
      print fnm,": ",id,"\n"
      yield id if block_given?
    end
  end

end

# check command line args
$0 = File.basename($0)
vars = {:datafile => "datafiles", :testfile => "testfiles",
  :verbose => false, :smooth => true}
parser = OptionParser.new do |q|
  q.banner = "Usage: #{$0} [options]\n"
  q.on_tail("-h", "--help", "show this message") {puts q; exit}
  q.on("-d", "--data=FILENAME", String,
    "data file names [default: 'datafiles']") {|vars[:datafile]|}
  q.on("-t", "--test=FILENAME", String,
    "test file name [default: 'testfiles']") {|vars[:testfile]|}
  q.on("-n", "--nosmooth", "do not smooth training data") {|vars[:nosmooth]|}
  q.on("-v", "--verbose", "be verbose") {|vars[:verbose]|}
end
begin
  parser.parse!(ARGV)
```

```
rescue OptionParser::ParseError
  STDERR.puts parser
  exit 1
end

begin
  c = Classifier.new(vars[:datafile], vars[:verbose])
  c.read_data
  c.smooth_data unless vars[:nosmooth]
  f = FileReader.new(vars[:testfile], c,  vars[:verbose])
  f.classify_input
  #f.classify_input {|res| print "==> ",res,"\n"}
rescue
  STDERR.puts "Something bad happened: #{$!}\n"
ensure
end
```

I.4.4 cserver.rb

Klassifikationsserver (s. C.1.4)

```
#!/usr/local/bin/ruby

# $Header: /opt/cvs/artus/cserver.rb,v 1.2 2001/03/23 13:10:03 koks-cvs Exp $

# cserver.rb - try to identify the language of a text (server version)
#
# KOKS project
# Arno Erpenbeck, 11.03.2001

require 'optparse'
require 'socket'
require 'classifier'

class SocketReader

  def initialize(socket, classifier, verbose=true)
    @socket = socket
    @classifier = classifier
    @verbose = verbose
  end

  # read test data from socket and classify
  def classify_input
    print "Reading test data from ",@socket,"...\n" if @verbose
    p = {}
    begin
      while @socket.gets
        if $_ =~ "END" ; break ; end
        $_.scan(/([a-z# ]+):(\d+)/) {|k, v| p.store(k, v)}
      end
      print "Read ",p.size," data pairs\n" if @verbose
    rescue
      STDERR.puts "Something bad happened: #{$!}\n"
    ensure
      #@socket.close
    end
    @classifier.classify(p)
  end

```

```
end

# check command line args
$0 = File.basename($0)
vars = {:datafile => "datafiles", :verbose => false, :nosmooth => false,
  :port => 9999}
parser = OptionParser.new do |q|
  q.banner = "Usage: #{$0} [options]\n"
  q.on_tail("-h", "--help", "show this message") {puts q; exit}
  q.on("-d", "--data=FILENAME", String,
    "data file names") {|vars[:datafile]|}
  q.on("-p", "--port=PORT", Integer,
    "listening port") {|vars[:port]|}
  q.on("-n", "--nosmooth",
    "do not smooth training data") {|vars[:nosmooth]|}
  q.on("-v", "--verbose",
    "be verbose") {|vars[:verbose]|}
end
begin
  parser.parse!(ARGV)
rescue OptionParser::ParseError
  STDERR.puts parser
  exit 1
end

begin
  c = Classifier.new(vars[:datafile], vars[:verbose])
  c.read_data
  c.smooth_data unless vars[:nosmooth]
  # open server socket
  gs = TCPServer.open("nulix.cl-ki.uni-osnabrueck.de", vars[:port])
  addr = gs.addr
  addr.shift
  print Time.now, ": cserver listening on ", addr.join(":"), "...\n"
  # wait for incoming connections
  while true do
    ns = gs.accept
    print Time.now, ": accepting connecting ", ns, "...\n"
    # start new thread for each connection
    Thread.start do
      socket = ns
      reader = SocketReader.new(socket, c)
      id = reader.classify_input
      # write out result
      socket.write(id)
      socket.close
      STDOUT.flush
    end
  end
rescue
  STDERR.puts "Something bad happened: #{$!}\n"
ensure
  gs.close
end
```

I.4.5 cclient.rb

Klassifikationsclient (s. C.1.4)

```
#!/usr/local/bin/ruby

```

```
# $Header: /opt/cvs/artus/cclient.rb,v 1.1 2001/03/12 14:14:12 koks-cvs Exp $

# cclient.rb - try to identify the language of a text (client version)
#
# KOKS project
# Arno Erpenbeck, 11.03.2001

require 'optparse'
require 'socket'

# check command line args
$0 = File.basename($0)
vars = {:testfile => "testfile", :server => "localhost", :port => 9999}
parser = OptionParser.new do |q|
  q.banner = "Usage: #{$0} [options]\n"
  q.on_tail("-h", "--help", "show this message") {puts q; exit}
  q.on("-t", "--test=FILENAME", String,
    "test file names [default: 'testfile']") {|vars[:testfile]|}
  q.on("-s", "--server=ADDRESS", String,
    "server address") {|vars[:server]|}
  q.on("-p", "--port=PORT", Integer,
    "server port") {|vars[:port]|}
end
begin
  parser.parse!(ARGV)
rescue OptionParser::ParseError
  STDERR.puts parser
  exit 1
end

begin
  # open socket to specified server
  socket = TCPSocket.open(vars[:server], vars[:port])
  # read in test file data and write to socket
  infile = open(vars[:testfile], "r")
  while infile.gets
    socket.write($_)
  end
  # signal end of data
  socket.write("END\n")
  socket.flush
  # wait for result
  id = socket.gets
  print "==> ",id,"\n"
rescue
  STDERR.puts "Something bad happened: #{$!}\n"
ensure
  socket.close
end
```

Anhang J

Wörterbuch-Code

J.1 Code für die Dictionary-Entry-Parser

J.1.1 Beispielcode für einfache Wörterbücher: vbsource.bas

Dieses Programm steht stellvertretend für alle Dictionary Entry Parser, welche Wörterbücher des Ëinfachen Fall”bearbeiteten.

```
Private Sub DateienBearbeiten()
’Norman Kummer 01-03 2001
’Bsp. für einen [D]ictionary[E]ntry[P]arser für Formate wie unten beschrieben.

’org.:(input)
’dt:: eng
’speziell abk.:
’    dt abk.:dt voll::ebg abk.:eng voll
’neu:(output)
’dt::eng##rest
’
’rest:
’    {dt};;{eng};;(dt);;(eng);;[dt];;[eng];;<dt>;;<eng>
’aFormat:
’    {d}info;;{e}info;;(d)info;;(e)info;;[d]info;;[e]info;;<d>info;;<e>info
’
’anmerkung:
’*manchmal in eng (;) verschiedene weitere Möglichkeiten z.B.:
’    jemanden davon abhalten; etwas zu tun :: to keep (prevent; stop) someone from doing something
’ im dt aber (;) erläuterungen!!!
’*möglicherweise letztes wort ersetzbar bei () -> also neue zeile
’*... lass ich weg, manchmal....
’*bei verben / -> kombinationen bilden
’*nachbearbeiten bei [alt], sonst vielleicht schwierigkeiten mit korpora
’ dafür neue einträge bilden, oder speziell abfragen beim parsen
’*kombinationen mit , z.B.:
’    - : Pfund (Gewicht) :: lb, lbs : pound, pounds
’* - bedeutet: keine info, oft’ - ’ auch ’ ? ’
’* fehler:
’    - Bin ich Jesus? :: AIJ : Bin ich Jesus?


’schritte:
’* suchen nach ’::’
’      - teilen in dt./eng.
’      - nur behandeln, wenn beide seiten mehr als ’ ’
’      - falls noch ’:’ vorhanden -> sonderbehandlung abk.
’* suchen auf beiden seiten nach ’;’
’      - neue zeilen draus machen + voller rest
```

```
'       - auf seiten alle klammern rausschmeißen
    Dim Datei As String
    Dim Textzeile As String
    Dim Ausgabezeile As String
    Dim Dt As String
    Dim Eng As String
    Dim DtEngFeld(1, 10) As String
    Dim RestAusgabe As String

    Dim SuchZeichen As String
    Dim Pos As Integer
    Dim AlterName As String
    Dim NeuerName As String
    Dim i As Integer
    Dim j As Integer
    Dim Pfad As String
    Pfad = "..\"
    Datei = "ger-eng.txt"

'öffnet Eingabedateien
    Open Pfad & Datei For Input As #1   ' Datei öffnen.
    FileLen (Pfad & Datei)
    ProgressBar1.Min = 0
    ProgressBar1.Value = ProgressBar1.Min
    ProgressBar1.Max = FileLen(Pfad & Datei)
'öffnet Ausgabedateien
    Open Pfad & "KOKSengdt_" & Datei For Output As #2
    Open Pfad & "KOKSdt_" & Datei For Output As #3
    Open Pfad & "KOKSeng_" & Datei For Output As #4
    Do While Not EOF(1) ' Schleife bis Dateiende.
        Line Input #1, Textzeile    ' Zeile in Variable einlesen.
        ProgressBar1.Value = ProgressBar1.Value + Len(Textzeile)
'* suchen nach '::' Delimiter1
        If Left(Textzeile, 1) <> "#" Then 'rem's ausschließen
            SuchZeichen = " :: "
            'Debug.Print Textzeile
            Pos = InStr(Textzeile, SuchZeichen)
            If Pos > 0 Then
    '       - teilen in dt./eng.
    '       - auf beiden seiten alle klammern rausschmeißen
                Dt = KeineKlammern(Left(Textzeile, Pos - 1))
                Eng = KeineKlammern(Right(Textzeile, Len(Textzeile) - Pos + 1 - Len(SuchZeichen)))
    '       - nur behandeln wenn beide seiten mehr als ' '
                If Dt <> " " And Eng <> " " Then
                    RestAusgabe = Textzeile
    '       - falls noch ':' vorhanden -> sonderbehandlung abk.(bei Wörterbuch 'Ding' vorhanden)
                    SuchZeichen = ":"
                    Pos = InStr(Dt, SuchZeichen) 'auch wenn eng. abk.
                    If Pos > 0 Then
                        'abk.
                    Else
    '* suchen auf beiden seiten nach ';' Delimiter2
                        Dim Seite(1) As String
                        Dim LetztePos As Integer
                        Dim Zähler As Integer
                        Dim Spezl As Integer
                        LetztePos = 1
                        Seite(0) = Dt 'bei i=0 für dt
                        Seite(1) = Eng 'bei i=1 für eng
                        Pos = 1
                        For i = 0 To 1
```

```
                            Zähler = 0
                            SuchZeichen = ";"
                            Spezl = -1
                            Do
    '       - neue zeilen draus machen + voller rest
                                'dt/eng felder aufbauen

                                SuchZeichen = ";"
                                Pos = InStr(LetztePos, Seite(i), SuchZeichen)
                                If Pos = 0 Then
                                    Pos = Len(Seite(i))
                                    Spezl = -1
                                Else
                                    Spezl = 0
                                End If
                                DtEngFeld(i, Zähler) = Mid(Seite(i), LetztePos, Pos - LetztePos - Spezl)
                                If Right(DtEngFeld(i, Zähler), 1) = " " Then
                                    DtEngFeld(i, Zähler) = Left(DtEngFeld(i, Zähler), Len(DtEngFeld(i, Zähler
                                End If
                                Zähler = Zähler + 1
                                LetztePos = Pos + 2
                                Spezl = -1
                            Loop Until Pos = Len(Seite(i))
                            Pos = 1
                            LetztePos = 1
                            'vorber
                        Next
                        'dt eng kombinieren und ausgeben
                        'neu:
                        'dt::eng##rest
                        For i = 0 To 10
                            If DtEngFeld(0, i) <> "" Then
                                For j = 0 To 10
                                    If DtEngFeld(1, j) <> "" Then
                                        If Right(DtEngFeld(0, i), 1) = " " Then
                                            DtEngFeld(0, i) = Left(DtEngFeld(0, i), Len(DtEngFeld(0, i)) - 1)
                                        End If
                                        If Right(DtEngFeld(1, j), 1) = " " Then
                                            DtEngFeld(1, j) = Left(DtEngFeld(1, j), Len(DtEngFeld(1, j)) - 1)
                                        End If
                                        Print #3, DtEngFeld(0, i) & "<KOKS>. <ABSATZ></KOKS>"
                                        Print #4, DtEngFeld(1, j) & "<KOKS>. <ABSATZ></KOKS>"
                                        Textzeile = DtEngFeld(0, i) & "::" & DtEngFeld(1, j) & "##" & RestAus
                                        If Left(RestAusgabe, 1) <> Left(Textzeile, 1) Then
                                            'Debug.Print Textzeile
                                            DoEvents
                                        End If
                                        Print #2, Textzeile
                                    Else
                                        j = 10
                                    End If
                                Next j
                            Else
                                i = 10
                            End If
                        Next i
                        For i = 0 To 10
                            For j = 0 To 10
                                DtEngFeld(0, i) = ""
                                DtEngFeld(1, j) = ""
                            Next
```

```
                        Next
                    End If
                End If
            End If
        End If
    Loop
'Dateien schließen
    Close #1
    Close #2
    Close #3
    Close #4
    MsgBox "Fertig!"
End Sub

Function KeineKlammern(Textzeile As String) As String
'löscht alle Klammern aus dem String 'Textzeile'
    Dim Klammern(3) As String
    Dim Pos As Integer
    Dim Pos2 As Integer
    Dim SuchZeichen As String
    Dim i As Integer
    Dim Textzeile2 As String
    Textzeile2 = Textzeile
    Klammern(0) = "()"
    Klammern(1) = "[]"
    Klammern(2) = "{}"
    Klammern(3) = "<>"
    For i = 0 To 3
        Pos = 1
        Do While Pos > 0
            SuchZeichen = Left(Klammern(i), 1)
            Pos = InStr(Textzeile2, SuchZeichen)
            SuchZeichen = Right(Klammern(i), 1)
            Pos2 = InStr(Textzeile2, SuchZeichen)

            If Pos > 0 Then
                Textzeile2 = Left(Textzeile2, Pos - 1) & Right(Textzeile2, Len(Textzeile2) - Pos2)
            End If
        Loop
    Next i
    If Right(Textzeile2, 1) = " " Then
        Textzeile2 = Left(Textzeile2, Len(Textzeile2) - 1)
    End If
    KeineKlammern = Textzeile2
End Function

Private Sub Command1_Click()
    DateienBearbeiten
End Sub
```

J.1.2 Code für das LQL-Wörterbuch

Diese Scripte wurden immer wieder angepaßt, bis das Lexikon die zum Einlesen benötigte Form hatte. Auch wurden vorhandene Fehler zusätzlich mit dem VI (Ersetzen/Löschen) ausgemerzt.

Schlüssel/werte aus Rohdaten aufsammeln: collect1.rb, collect2.rb, collect3.rb

```
#!/usr/local/bin/ruby

if ARGV.size == 1
        infile = open(ARGV[0], "r")
else
        infile = STDIN
end

keys = []

while line = infile.gets
        if line =~ /(^ |\()([a-z]+):/
                #puts $1
                keys << $2 unless keys.include?($2)
        end
end

infile.close if infile != STDIN

skeys = keys.sort
skeys.each {|idx| puts idx }
puts skeys.size
```

———-

```
#!/usr/local/bin/ruby

if ARGV.size == 1
        infile = open(ARGV[0], "r")
else
        infile = STDIN
end

keys = []

while line = infile.gets
        line.scan(/( |:)([A-ZÄÖÜßäöüßa-z_-]+):/) { |match|
        #line.scan(/([A-ZÄÖÜßäöüßa-z_]+)\(/) { |match|
                keys << match[1] unless keys.include?(match[1])
        }
end

infile.close if infile != STDIN

skeys = keys.sort
skeys.each {|idx| puts idx }
puts skeys.size
```

———-

```
#!/usr/local/bin/ruby

if ARGV.size == 1
        infile = open(ARGV[0], "r")
else
```

```
        infile = STDIN
end

keys = []

while line = infile.gets
        if line =~ /usage_note:([\w ]+)/
                puts $1
        end
        #line.scan(/usage_note:([A-Za-z ]+\()/) { |match|
                #print match
                #keys << match unless keys.include?(match)
        #}
end

infile.close if infile != STDIN

skeys = keys.sort
skeys.each {|idx| puts idx }
puts skeys.size
```

”collocat” Phrasen aus Lexikon aufsammeln: collocat.rb

```
#!/usr/local/bin/ruby

=begin
 collocat.rb
 'collocat'-Phrasen aufsammeln
 Norman Kummer, Arno Erpenbeck
 KOKS Projekt, April/Mai 01
=end

require 'string-utils'

if ARGV.size == 1
        infile = open(ARGV[0], "r")
else
        infile = STDIN
end

class Phrase
        attr_accessor :key

        def initialize(key=nil)
                key = "" unless key
                @key = key
                @source = []
                @target = []
                @current = 0
        end

        def reset_all
                @source = []
                @target = []
        end

        def reset
                @target[@current] = []
        end

        def source=(what)
                @source << what
        end

        def target=(what)
                @target[@current] << what
        end

        def fix_target
                @target[@current].last << ")" unless @target[@current].last[-1,1] == ')'
        end

        def proceed
                @current += 1
        end

        def dump
                print @key, "\t"
                @source.each {|x| print x, " "}
                print "\t"
                @target[@current].each {|x| print x, " "}
                print "\n"
```

```
60          end
61
62  end
63
64  lines = 0
65
66  # jede Zeile lesen und auseinander nehmen
67  while line = infile.gets
68          lines += 1
69          STDERR.print "." if lines % 1000 == 0
70          modus = :egal # Modus
71          found = false # Deutsches Wort gefunden
72          count = 0 # Klammerzaehler
73          saved = 0
74          extra = 0
75          phrase = Phrase.new
76          # alle Tokens in einer Zeile anschauen
77          line.split.each { |token|
78                  count += token.count_braces
79                  # erstes Token ist deutsch
80                  if !found
81                          phrase.key = token
82                          found = true
83                  # wenn 'collocat(' kommt wird es spannend
84                  elsif token == "collocat("
85                          modus = :coll
86                  # 'source:' gefunden -> weiterlaufen und aufsammeln
87                  elsif modus == :coll && token == "source:"
88                          modus = :source
89                          phrase.reset_all
90                  # aufsammeln bis zum key:
91                  elsif modus == :source
92                          if token.is_key?
93                                  modus = :skip
94                          elsif token == "targ("
95                                  modus = :targ
96                          else
97                                  phrase.source = token
98                          end
99                  elsif modus == :skip && token == "targ("
100                         modus = :targ
101                 elsif modus == :targ && token == "target("
102                         modus = :target
103                         saved = count
104                 elsif modus == :target
105                         if token == "word:" || token == "phrase:"
106                                 modus = :word
107                                 extra = 0
108                                 phrase.reset
109                         end
110                 elsif modus == :word
111                         # zu weit gelaufen -> modus zurück setzen
112                         if token.is_key? || count < saved
113                                 phrase.target = token.clear(extra) if token.is_end?
114                                 phrase.fix_target if extra > 0
115                                 phrase.dump
116                                 phrase.proceed
117                                 modus = :skip
118                         else
119                                 # Klammergebirge nachher reparieren
120                                 if token.is_start? && !token.is_end?
```

```
                                        extra = 1
                                    elsif token.is_end?
                                        extra = 0
                                    end
                                    phrase.target = token
                                end
                            end
                        }
end

infile.close if infile != STDIN
```

’ ’ in Phrasen durch Schlüsselwort ersetzen: ersetzer.rb

```
#!/usr/local/bin/ruby

=begin
  ersetzer.rb
 ’~’ in Phrasen durch Wort ersetzen
 Norman Kummer, Arno Erpenbeck
 KOKS Projekt, April/Mai 01
=end

class String

        def replace(str)
                pos = self.index(’~’)
                if !pos
                        # kein ’~’ in Wort
                        return self
                elsif pos == 0
                        # ’~’ am Wortanfang -> ersetzen
                        self.gsub(/~/, str)
                else
                        # ’~’ nicht am Wortanfang -> kopieren und Teil ersetzen
                        pre = self[0, pos]
                        post = self[pos, self.length].gsub(/~/, str[pos, str.length])
                        pre << post
                end
        end

end

# open input and output files
if ARGV.length != 3
        STDERR.puts "Usage: ersetzer <lex-file> <german-out> <english-out>"
        exit 1
end
if ARGV[0] == "-"
        infile = STDIN
else
        infile = open(ARGV[0], "r")
end
gfile = open(ARGV[1], "w")
efile = open(ARGV[2], "w")

# Zeilen verarbeiten, dabei ’~’ durch keyword ersetzen
lines = 0
while line = infile.gets
        lines += 1
        tokens = line.split(/\t/)
        if tokens.size != 3
                STDERR.print("Illegal line: #{lines}\n")
                #exit 2
                next
        end
        german = tokens[0]
        tokens[1].split.each {|token| gfile.print token.strip.replace(german), " " }
        gfile.print "\n"
        tokens[2].each {|token| efile.print token.strip, " "}
        efile.print "\n"
end

```

```
60 # aufraeumen
61 if infile != STDIN
62         infile.close
63 end
64 gfile.close
65 efile.close
```

Übersetzungspaare aufsammeln (eng-ger): etran.rb

```
#!/usr/local/bin/ruby

=begin
 etran.rb
 'tran'-Phrasen aufsammeln (eng-ger)
 Norman Kummer, Arno Erpenbeck
 KOKS Projekt, April/Mai 2001
=end

require 'string-utils'
require 'translation'

if ARGV.size == 1
        infile = open(ARGV[0], "r")
else
        infile = STDIN
end

lines = 0
# jede Zeile lesen und auseinander nehmen
while line = infile.gets
        lines += 1
        STDERR.print "." if lines % 1000 == 0
        modus = :egal # Modus
        count = 0 # Klammerzaehler
        extra = 0
        saved = 0
        trans = Translation.new
        # alle Tokens in einer Zeile anschauen
        line.split.each {|token|
                count += token.count_braces
                # solange englisch, bis '^('
                if token == '^('
                        modus = :skip
                elsif modus == :egal
                        trans.orig = token
                # wenn 'hom(' kommt auf 'pos:' warten
                elsif token == "hom("
                        modus = :hom
                        trans.reset_pos
                # wenn 'tran(' kommt wird es spannend
                elsif token == "tran("
                        modus = :tran
                        saved = count
                # 'pos:' gefunden -> Wortart aufsammeln
                elsif modus == :hom && token == "pos:"
                        modus = :pos
                # ersten Wert nach 'pos:' behalten
                elsif modus == :pos
                        if token.is_key? || token.is_brace?
                                modus = :skip
                        else
                                trans.pos = token
                        end
                # 'word:' gefunden -> weiterlaufen und aufsammeln
                elsif modus == :tran && token == "word:"
                        modus = :word
                        extra = 0
                        trans.reset_tran
```

```
60                  # 'phrase:' gefunden -> weiterlaufen und aufsammeln
61                  elsif modus == :tran && token == "phrase:"
62                          modus = :phrase
63                  # aufsammeln bis zum Ende
64                  elsif modus == :word
65                          # zu weit gelaufen -> modus zurück setzen
66                          if token.is_key? || token.is_brace? || count < saved
67                                  trans.tran = token.clear(extra) if token.is_end?
68                                  trans.fix_tran if extra > 0
69                                  trans.dump
70                                  modus = :skip
71                          else
72                                  # Klammergebirge nachher reparieren
73                                  if token.is_start? && token.count_braces > 0 && !token.is_end?
74                                          extra = 1
75                                  elsif token.is_end?
76                                          extra = 0
77                                  end
78                                  trans.tran = token
79                          end
80                  end
81          }
82  end
83
84  infile.close if infile != STDIN
85
```

Übersetzungspaare aufsammeln (ger-eng): gtran.rb

```
#!/usr/local/bin/ruby

=begin
 gtran.rb
 'tran'-Phrasen aufsammeln (ger-eng)
 Norman Kummer, Arno Erpenbeck
 KOKS Projekt, April/Mai 2001
=end

require 'string-utils'
require 'translation'

if ARGV.size == 1
        infile = open(ARGV[0], "r")
else
        infile = STDIN
end

lines = 0

# jede Zeile lesen und auseinander nehmen
while line = infile.gets
        lines += 1
        STDERR.print "." if lines % 1000 == 0
        modus = :egal # Modus
        found = false # Deutsches Wort gefunden
        count = 0 # Klammerzaehler
        extra = 0
        saved = 0
        trans = Translation.new
        # alle Tokens in einer Zeile anschauen
        line.split.each {|token|
                count += token.count_braces
                # erstes Token ist deutsch
                if !found
                        trans.orig = token
                        found = true
                # wenn 'hom(' kommt auf 'pos:' warten
                elsif token == "hom("
                        modus = :hom
                        trans.reset_pos
                # wenn 'tran(' kommt wird es spannend
                elsif token == "tran("
                        modus = :tran
                        saved = count
                # 'pos:' gefunden -> Wortart aufsammeln
                elsif modus == :hom && token == "pos:"
                        modus = :pos
                # ersten Wert nach 'pos:' behalten
                elsif modus == :pos
                        if token.is_key? || token.is_brace?
                                modus = :egal
                        else
                                trans.pos = token
                        end
                # 'word:' gefunden -> weiterlaufen und aufsammeln
                elsif modus == :tran && token == "word:"
                        modus = :word
                        extra = 0
```

```
60                      trans.reset_tran
61              # 'phrase:' gefunden -> weiterlaufen und aufsammeln
62              elsif modus == :tran && token == "phrase:"
63                      modus = :phrase
64              # aufsammeln bis zum Ende
65              elsif modus == :word
66                      # zu weit gelaufen -> modus zurück setzen
67                      if token.is_key? || token.is_brace? || count < saved
68                              trans.tran = token.clear(extra) if token.is_end?
69                              trans.fix_tran if extra > 0
70                              trans.dump
71                              modus = :egal
72                      else
73                              # Klammergebirge nachher reparieren
74                              if token.is_start? && !token.is_end?
75                                      extra = 1
76                              elsif token.is_end?
77                                      extra = 0
78                              end
79                              trans.tran = token
80                      end
81              end
82      }
83  end
84
85  infile.close if infile != STDIN
```

Tags von LQL auf IMS abbilden: mapper.rb

Das LQL-Wörterbuch hatte eigene Taginformationen, welche mit diesem Script auf das IMS-Set gemappt wurden.

```
#!/usr/local/bin/ruby

=begin
 mapper.rb
 Tags von LQL auf eigenes Tagset abbilden
 Arno Erpenbeck, Norman Kummer
 KOKS Projekt, April/Mai 2001
=end

# ? noch diskutieren
# ! von Hand ändern
# "" löschen

hash = {
"" => "unknown",
"adj" => "adj",
"adj attr usu superl" => "adj",
"adj usu attr" => "adj",
"adj usu pred" => "adj",
"addj pred" => "adj",
"adj pred, adv" => "adj",
"adj pred, adv lines, ropes" => "adj",
"adj suf" => "adj",
"adj usu attr happiness" => "adj",
"adj, interj" => "adj",
"adv" => "adv",
"adv see adj" => "adv",
"adv suf" => "adv",
"adv, interj" => "adv",
"conj" => "kon",
"defective vb" => "unknown",
"f or" => "nn",
"imper" => "imp",
"in" => "unknown",
"in cpds" => "unknown",
"in cpds mit adj" => "unknown",
"in cpds mit superl" => "unknown",
"in cpds gun, car, soldier" => "unknown",
"in cpds post, town, zone" => "unknown",
"indef art" => "art",
"interj" => "itj",
"interj fire" => "itj",
"interj old" => "itj",
"interrog adj inv" => "unknown",
"interrog adv" => "unknown",
"interrog, rel adv" => "unknown",
"m" => "nn",
"m or" => "nn",
"n" => "nn",
"n see adj" => "nn",
"n also" => "nn",
"n inf see adj" => "nn",
"n pl" => "nn",
"n see adj" => "nn",
"n sing or pl" => "nn",
"n usu pl" => "nn",
"n, interj" => "nn",
```

```
"n, pron, adv" => "nn",
"no art" => "nn",
"no art, inv" => "nn",
"nt" => "nn",
"nt or" => "nn",
"pl" => "unknown",
"poss adj" => "adj",
"pref" => "unknown",
"pref brother, mother etc" => "unknown",
"pref in cpd vbs" => "unknown",
"pref mit n" => "unknown",
"prep" => "prep",
"pret, ptp" => "unknown",
"pret, ptp ~ vt" => "unknown",
"pron" => "pron",
"rel adv" => "unknown",
"suf" => "unknown",
"v" => "v",
"v aux" => "va",
"vb suf" => "v",
"vi" => "vi",
"vi, vt sep" => "vi",
"vt always separate" => "vt",
"vt sep and insep" => "vt",
"vt sep auch irreg" => "vt",
"vt usu pass" => "vt",
"vt, vt impers" => "vt",
"weak form" => "unknown"}

if ARGV.length != 3
        STDERR.puts "Usage: mapper <lex-file> <german-out> <english-out>"
        exit 1
end
if ARGV[0] == "-"
        infile = STDIN
else
        infile = open(ARGV[0], "r")
end
gfile = open(ARGV[1], "w")
efile = open(ARGV[2], "w")

count = 0
stat = {}
while line = infile.gets
        count += 1
        tokens = line.split('\t')
        if tokens.length != 3
                STDERR.puts "Illegal line #{count}"
        end
        lemma = tokens[0]
        tag = tokens[1]
        if lemma[-1,1] == '-' || lemma[0,1] == '-'
                STDERR.puts "Deleting token '#{lemma}'"
                next
        end
        if !hash.has_key?(tag)
                STDERR.puts "Illegal Tag '#{tag}' in line #{count}"
                next
        end
        pos = hash[tag]
        if !stat.has_key?(pos)
```

```
                stat[pos] = 0
        end
        stat[pos] += 1
        tran = tokens[2]
        gfile.print lemma,"\t",pos,"\t",lemma,"\n"
        efile.print tran.strip,"\n"
end

infile.close unless infile == STDIN
stat.sort.each {|k, v| puts "#{k}: #{v}"}
```

Diverse Stringmethoden: string-utils.rb

```
=begin
 string-utils.rb
 div. String-Methoden
 Arno Erpenbeck
 KOKS Projekt, April/Mai 2001
=end

class String
        def is_key?
                return self[-1,1] == ':'
        end

        def is_brace?
                return self[-1,1] == '('
        end

        def is_end?
                return self[-1,1] == ')'
        end

        def is_start?
                return self[0,1] == '('
        end

        def count_braces
                count = 0
                (0..self.length).each { |idx|
                        if self[idx,1] == '('
                                count += 1
                        elsif self[idx,1] == ')'
                                count -= 1
                        end
                }
                return count
        end

        def clear(extra = 0)
                del = self.count_braces
                if del >= 0
                        copy = self
                else
                        del += extra
                        copy = self[0,self.length+del]
                end
                return copy
        end
end
```

Kapselt Übersetzungsdaten aus Lexikon: translation.rb

```
=begin
 translation.rb
 kapselt Übersetzungsdaten aus Lexikon
 Arno Erpenbeck
 KOKS Projekt, April/Mai 2001
=end

class Translation

        def initialize(orig=nil)
                @orig = []
                @orig << orig if orig
                @pos = []
                @tran = []
        end

        def reset_pos
                @pos = []
        end

        def reset_tran
                @tran = []
        end

        def fix_tran
                @tran.last << ")" unless @tran.last[-1,1] == ')'
        end

        def orig=(what)
                @orig << what
        end

        def pos=(what)
                @pos << what
        end

        def tran=(what)
                @tran << what
        end

        def dump
                @orig.each {|x| print x, " "}
                print "[ "
                @pos.each {|x| print x, " "}
                print "] "
                @tran.each {|x| print x, " "}
                print "\n"
        end
end


```

Anhang K

Normalisierung – Code

K.1 `normalize.py`

Verwandelt Dateien aus vielen verschiedenen Formaten ins KoKS-Format.

```
#!/usr/bin/python
from htmllib import *
from sgmllib import *
from formatter import *
import sys, string, os, re

#TODO: bei SGML $xxx wegschmeissen ???
#TODO: Satzzeichen bei SGML!

class ToAscii:

    def __init__(self, file, format=""):
        self.infile = file
        if format:
            (self.outfile, self.done) = getfile(self.infile, format)
        else:
            self.outfile = file
            self.done = 1

    def convert(self):
        return self.outfile

class Html2Ascii(ToAscii, HTMLParser):

    def __init__(self, file):
        ToAscii.__init__(self, file, "asc")
        if not self.done:
            self.fd = open(self.outfile, "w")
            HTMLParser.__init__(self, AbstractFormatter(DumbWriter(self.fd)))

    def convert(self):
        if not self.done:
            sys.stderr.write("Html2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
            HTMLParser.feed(self, open(self.infile).read())
            self.close()
            self.fd.close()
        return self.outfile

    def anchor_end(self):
        pass

    def do_hr(self, attrs):
```

```
43          self.formatter.end_paragraph(2)
44
45      def do_img(self, attrs):
46          pass
47
48      def do_li(self, attrs):
49          HTMLParser.do_li(self, attrs)
50          self.formatter.add_line_break()
51
52      def do_p(self, attrs):
53          HTMLParser.do_p(self, attrs)
54          self.formatter.writer.send_literal_data("\n<ABSATZ>\n")
55
56  class Sgml2Ascii(ToAscii, SGMLParser):
57      #TODO: $xxx wegschmeissen ???
58      #TODO: Satzzeichen!!!
59
60      def __init__(self, file):
61          ToAscii.__init__(self, file, "asc")
62          if not self.done:
63              SGMLParser.__init__(self)
64          self.has_content = 0
65          self.has_dot = 0
66          self.is_head = 0
67          self.punktre = re.compile('.*[!\?:\.]$', re.DOTALL)
68
69      def handle_data(self, data):
70          data = string.strip(data)
71          if data:
72              if self.is_head:
73                  self.out.write("\n")
74              self.out.write(data + " ")
75              self.has_content = 1
76              if self.punktre.match(data):
77                  self.has_dot = 1
78              elif self.is_head:
79                  self.out.write(".\n")
80                  self.has_dot = 1
81
82  #    def start_head(self, args):
83  #        pass
84
85  #    def end_head(self):
86  #        self.out.write("\n<ABSATZ>\n")
87
88      def start_s(self, args):
89          self.has_dot = 0
90
91      def end_s(self):
92          if not self.has_dot:
93              self.out.write(" .\n")
94          self.has_content = 1
95
96      def start_p(self, args):
97          self.has_dot = 0
98          self.out.write('\n')
99          if ('type','head') in args:
100             self.is_head = 1
101
102     def end_p(self):
103         self.is_head = 0
```

```
104
105     def start_h1(self, args):
106         if self.has_content:      # kein Delimiter vor erster Ueberschrift
107             self.out.write("\n<ABSATZ>\n")
108
109     def end_h1(self):
110         self.out.write("\n<ABSATZ>\n")
111         self.has_content = 0
112
113     def convert(self):
114         if not self.done:
115             sys.stderr.write("Sgml2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
116             self.out = open(self.outfile, "w")
117             SGMLParser.feed(self, open(self.infile).read())
118             self.out.close()
119             self.close()
120         return self.outfile
121
122 class Pdf2Ascii(ToAscii):
123
124     def __init__(self, file):
125         ToAscii.__init__(self, file, "html")
126
127     def convert(self):
128         if not self.done:
129             sys.stderr.write("Pdf2Ascii:  converting %s -> %s\n" %(self.infile, self.outfile))
130             os.system("/usr/local/bin/pdf2html '%s' '%s'" %(self.infile, self.outfile))
131             #sys.stderr.write("Pdf2Ascii: converting %s from html...\n" %self.outfile)
132             h2a = Html2Ascii(self.outfile)
133             ascfile = h2a.convert()
134             return ascfile
135         return self.outfile
136
137 class Ps2Ascii(ToAscii):
138
139     def __init__(self, file):
140         ToAscii.__init__(self, file, "asc")
141
142     def convert(self):
143         if not self.done:
144             sys.stderr.write("Ps2Ascii:   converting %s -> %s\n" %(self.infile, self.outfile))
145             os.system("/usr/bin/ps2ascii '%s' '%s'" %(self.infile, self.outfile))
146         return self.outfile
147
148 class Word2Ascii:
149
150     def __init__(self, file):
151         ToAscii.__init__(self, file, "asc")
152
153     def convert(self):
154         if not self.done:
155             sys.stderr.write("Word2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
156             os.system("/usr/bin/catdoc -a '%s' '%s'" %(self.infile, self.outfile))
157         return self.outfile
158
159 class Asc2Asc(ToAscii):
160     def __init__(self, file):
161         ToAscii.__init__(self, file, "asc2")
162
163     def convert(self):
164         if not self.done:
```

```
165             sys.stderr.write("Asc2Ascii:  converting %s -> %s\n" %(self.infile, self.outfile))
166             input = open(self.infile, "r")
167             output = open(self.outfile, "w")
168 #            nlcount = 0
169 #            emptyline = 0
170 #            absatz = 0
171             for line in input.readlines():
172 #                if string.strip(line) == "":
173 #                    emptyline = 1
174 #                else:
175 #                    emptyline = 0
176 #                if string.find(line, "<ABSATZ>") >= 0:
177 #                    absatz = 1
178 #                if not emptyline:
179 #                    if nlcount:
180 #                        if not absatz:
181 #                            output.write("\n<ABSATZ>\n")
182 #                        nlcount = 0
183 #                        absatz = 0
184 #                else:
185 #                    nlcount = nlcount + 1
186 
187                 # convert to regular quotes
188                 line = re.sub("<<", "\"", line)
189                 line = re.sub(">>", "\"", line)
190                 output.write(line)
191             # Das scheint irgendwie falsch
192             #if line[-9:] != "<ABSATZ>\n":            # keine leeren Absaetze erzeugen
193             #    output.write("\n<ABSATZ>\n")
194             # tree-tagger-bug - workaround
195             output.write("\n<dokende>\nDas Dokument ist nun wirklich zu Ende.\n")
196             output.close()
197             input.close()
198         return self.outfile
199 
200 def conv2ascii(file, format):
201     c2a = None
202     if format == "html":
203         c2a = Html2Ascii(file)
204     elif format == "sgml":
205         c2a = Sgml2Ascii(file)
206     elif format == "pdf":
207         c2a = Pdf2Ascii(file)
208     elif format == "ps":
209         c2a = Ps2Ascii(file)
210     elif format == "doc":
211         c2a = Word2Ascii(file)
212     elif format == "plain":
213         c2a = ToAscii(file)
214     else:
215         return None
216     a2a = Asc2Asc(c2a.convert())
217     return a2a.convert()
218 #    return c2a.convert()
219 
220 def getfile(file, end, cutoff=1):
221     if(cutoff):
222         newfile = file[:string.rfind(file, ".")] + "." + end
223     else:
224         newfile = file + "." + end
225     done = 0
```

```
    try:
        open(newfile)
        done = 1
    except:
        pass
    return (newfile, done)

def main():
    if len(sys.argv) < 3:
        print "Usage: normalize.py <file> <format>"
        print "       <format> is one of html,sgml,pdf,ps,doc,plain"
        sys.exit(1)
    conv2ascii(sys.argv[1], sys.argv[2])

if __name__ == '__main__':
    main()
```

K.2 preproc.py

Normalisiert Korpora und trägt sie in die Datenbank ein.

```
#!/usr/bin/python

# $Header: /opt/cvs/bin/preproc.py,v 1.48 2001/08/05 14:38:35 koks Exp $
#
# preproc.py

from xmllib import *
import sys, getopt, string, os
from DatabaseAPI import config
import korpus2db, normalize

class SystemCall:

    def __init__(self, file, end, cutoff=1):
        self.infile = file
        (self.outfile, self.done) = normalize.getfile(self.infile, end, cutoff)

    def call(self, cmd, args, name):
        if not self.done:
            sys.stderr.write("%s %s\n" %(name, self.infile))
            print cmd, args
            os.system(cmd + " " + args)
        return self.outfile

class InsertDelimiters:

    def __init__(self, file):
        self.infile = file
        (self.outfile, self.done) = normalize.getfile(self.infile, "del")

    def insdels(self, soft, sign):
        if not self.done:
            print "inserting delimiters..."
            file = open(self.infile, "r")
            out = open(self.outfile, "w")
            for line in file.readlines():
                delim_written = 0
                if string.find(line, sign) > -1:
```

```
                    out.write(line)
                    out.write(soft)
                elif string.find(line, "<ABSATZ>") > -1 and not delim_written:
                    out.write(soft)
                    out.write(line)
                else:
                    out.write(line)
                    delim_written = 0
            file.close()
            out.close()
        return self.outfile


class MapParser(XMLParser):

    def __init__(self, file, kind):
        XMLParser.__init__(self)
        self.file = file
        if string.rfind(file, "/") > -1:
            self.dir = file[:string.rfind(file, "/")] + "/"
        else:
            self.dir = ""
        self.use = 0
        self.kind = kind


    def parse_mapping(self):
        self.mapping = []
        self.nextdata = None
        print "parsing index file " + self.file
        self.feed(open(self.file).read())
        self.close()

    def start_mapping(self, attrs):
        pass

    def end_mapping(self):
        pass

    def start_document(self, attrs):
        self.idx = len(self.mapping)
        if not attrs.has_key("use") or attrs["use"] == "yes":
            self.use = 1
        else:
            self.use = 0
        if self.kind and attrs.has_key("kind") and attrs["kind"] != self.kind:
            self.use = 0
        if self.use:
           self.mapping.append({})

    def end_document(self):
        self.end_info()

    def start_part(self, attrs):
        if self.use:
            self.lang = attrs["lang"]
            self.mapping[self.idx][self.lang] = {}
            tagset = "ims"
            if attrs.has_key("tagset"):
                tagset = attrs["tagset"]
            self.mapping[self.idx][self.lang]["tagset"] = "tagset_" + tagset
```

```
100
101
102     def end_part(self):
103         pass
104
105     def start_file(self, attrs):
106         pass
107
108     def end_file(self):
109         pass
110
111     def start_filename(self, attrs):
112         if self.use:
113             self.nextdata = "file"
114             self.mapping[self.idx][self.lang]["format"] = attrs["format"]
115
116     def end_filename(self):
117         self.nextdata = None
118
119     def start_info(self, attrs):
120         pass
121
122     def end_info(self):
123         if self.use:
124             for key in ['author', 'comment', 'date', 'source', 'sourcetype']:
125                 if not self.mapping[self.idx].has_key(key):
126                     self.mapping[self.idx][key] = '?'
127
128     def start_author(self, attrs):
129         if self.use:
130             self.nextdata = "author"
131
132     def end_author(self):
133         self.nextdata = None
134
135     def start_comment(self, attrs):
136         if self.use:
137             self.nextdata = "comment"
138
139     def end_comment(self):
140         self.nextdata = None
141
142     def start_date(self, attrs):
143         if self.use:
144             self.nextdata = "date"
145
146     def end_date(self):
147         self.nextdata = None
148
149     def start_source(self, attrs):
150         if self.use:
151             self.mapping[self.idx]["sourcetype"] = attrs["type"]
152             self.nextdata = "source"
153
154     def end_source(self):
155         self.nextdata = None
156
157     def handle_data(self, data):
158         if self.use:
159             value = string.strip(data)
160             if not value:
```

```
                value = "?"
            if self.nextdata == "comment":
                self.mapping[self.idx]["comment"] = value
            elif self.nextdata == "file":
                self.mapping[self.idx][self.lang]["file"] = self.dir + value
            elif self.nextdata == "author":
                self.mapping[self.idx]["author"] = value
            elif self.nextdata == "date":
                self.mapping[self.idx]["date"] = value
            elif self.nextdata == "source":
                self.mapping[self.idx]["source"] = value

class Enter2Db:

    def __init__(self):
        self.kdb = korpus2db.Korpus()
        self.langmap = {'de': 1, 'en' : 2}

    def enter2db(self, info):
        file_de = info['de']['tagfile'] + ".al"
        file_en = info['en']['tagfile'] + ".al"
        (f1, done1) = normalize.getfile(file_de, "db")
        (f2, done2) = normalize.getfile(file_en, "db")
        if done1 or done2:
            print "texts %s %s already entered into db..." %(file_en, file_de)
            self.kdb.close()
            return
        qid = self.kdb.enterquelle(info['source'], info['sourcetype'], info['comment'])
        herk = self.kdb.enterherkunft(info['de']['file'], info['author'], info['date'], qid)
        print qid, herk
        try:
            print "entering texts into database..."
            self.kdb.txt2db(herk, (file_de, "de", info['de']["tagset"]), (file_en, "en", info['en']["tagset"]))
            self.kdb.close()
            for i in (f1, f2):
                file = open(i, "w")
                file.close()
        except IOError:
            print "no aligned files found!"
#        except:
#            print "something bad happened while trying to enter %s" %info

def usage():
    print "Usage: preproc.py [-c|-n] [-a] [-i <id>] [-k <kind>] <xml-files> ..."
    print "-a        skip alignment"
    print "-c        only convert"
    print "-i <id>   run under different id (use with care)"
    print "-k <kind> only process entries of specified kind"
    print "-n        do not enter stuff into database"
    sys.exit()

def main():
    # process command line args
    try:
        opts, args = getopt.getopt(sys.argv[1:], "achni:k:", ["align", "convert", "help", "nodb", "id=", "kind="
    except getopt.GetoptError:
        # print help information and exit:
        usage()
    opt_id = None
    opt_kind = None
    opt_align = 1
```

```
222     opt_convert = 0
223     opt_enterdb = 1
224     for o, a in opts:
225         if o in ("-a", "--align"):
226             opt_align = 0
227         if o in ("-c", "--convert"):
228             if not opt_enterdb:
229                 usage()
230             opt_convert = 1
231         if o in ("-n", "--nodb"):
232             if opt_convert:
233                 usage()
234             opt_enterdb = 0
235         if o in ("-h", "--help"):
236             usage()
237         if o in ("-i", "--id"):
238             opt_id = a
239         if o in ("-k", "--kind"):
240             opt_kind = a
241     if len(args) < 1:
242         usage()
243 
244     # flag -i should only be used together with -n
245     if opt_id and opt_enterdb:
246         sys.stderr.write("Warning: Using different id than dbname!\n")
247 
248     # setup runfile name
249     if opt_id:
250         runfile = "/tmp/preproc_" + opt_id + ".running"
251     else:
252         runfile = "/tmp/preproc_" + config.dbname + ".running"
253 
254     # check if already running
255     try:
256         file = open(runfile, "r")
257         file.close()
258         sys.exit("preproc.py is already running!")
259     except IOError:
260         pass
261     file = open(runfile, "w")
262     file.close()
263 
264     langmap = {"en" : "english", "de": "german"}
265 
266     #for i in sys.argv[start:]:
267     al_empty_err = []
268     for i in args:
269         xmlp = MapParser(i, opt_kind)
270         xmlp.parse_mapping()
271         for map in xmlp.mapping:
272             for lang in ["de", "en"]: #{
273                 if not map.has_key(lang):
274                     break
275                 ldict = map[lang]
276                 file = ldict["file"]
277                 if not os.access(file, os.R_OK):
278                     sys.stderr.write("File not found: %s\n" %file)
279                     break
280                 format = ldict["format"]
281                 if format != "tagged":
282                     ascfile = normalize.conv2ascii(file, format)
```

```
                    if opt_convert:
                        continue
                    if (ascfile):
                        a2t = SystemCall(ascfile, "tag")
                        pretagfile = a2t.call(getattr(config, "tagger_" + lang), " '%s' > '%s'" %(a2t.infile, a2
                        ldict["tagfile"] = pretagfile
                        #if(pretagfile):
                            # insdel = InsertDelimiters(pretagfile)
                            # ldict['tagfile'] = insdel.insdels("<SATZ>\n<segmentgrenze>\n", langmap[lang + "_si
                            # insdel = SystemCall(pretagfile, "del")
                            # ldict['tagfile'] = insdel.call("punkt-tagger-%s" %langmap[lang],
                            #                                " < '%s' > '%s' " %(insdel.infile, insdel.outfile),
                            #                                "inserting delimiters into ")
                else:
                    if opt_convert:
                        sys.stderr.write("%s: nothing to do, because format = %s\n" %(file, format))
                        break # ?continue?
                    ldict["tagfile"] = ldict["file"]
                sys.stderr.flush()
            if(map.has_key('de') and map.has_key('en')): #{
                de_file = None
                if  map['de'].has_key('tagfile'):
                  de_file = map['de']['tagfile']
                en_file = None
                if  map['en'].has_key('tagfile'):
                  en_file = map['en']['tagfile']
                if de_file and en_file and opt_align:
                    al = SystemCall(de_file, "al", 0)
                    #al.call(config.align, " -e -D '<dokende>' -d '<ABSATZ>' '%s' '%s'" %(de_file, en_file), "al
                    al.call(config.align, " -e -D '<ABSATZ>' -d '<segmentgrenze>' '%s' '%s'" %(de_file, en_file)
                    if opt_enterdb:
                        quit = 0
                        for f in(de_file, en_file):
                            al_file = f + ".al"
                            if not os.access(al_file, os.R_OK) or \
                               not string.strip(open(al_file, "r").read()):
                                #al_file.close()
                                al_empty_err.append(f)
                                quit = 1
                        if quit:
                            continue
                        print "enter2db"
                        e2d = Enter2Db()
                        e2d.enter2db(map)
                sys.stderr.flush()
            #}
    if al_empty_err:
        print "!!!empty aligned files:", al_empty_err, "!!!"
    os.unlink(runfile)

main()
```

K.3 py2pl.py

Macht die Python-Konfigurationsdateien für Perl nutzbar.

```
#!/usr/bin/python

# $Header: /opt/cvs/bin/py2pl.py,v 1.7 2001/08/04 11:58:40 koks-cvs Exp $
#
```

```
# read config.py and write out data in perl-usable style
#
# KOKS project
# 31.07.2001

from DatabaseAPI import config
import sys

# TODO
# - fix error with arrays/hashes

print """
#!/usr/bin/perl

# config file generated by py2pl

use strict;
use File::stat;

# testen, ob config.py neuer ist als config.pl
my $dir = ‘dirname $0‘;
chomp($dir);
my $pyfile;
if (defined(%ENV->{PYTHONPATH})) {
  my @paths = split/:/, %ENV->{PYTHONPATH};
  my $path = "";
  foreach $path (@paths) {
    $pyfile = $path . "/DatabaseAPI/config.py";
    open FOO, $path and last;
  }
  close FOO;
} else {
  $pyfile = $dir . "/config.py";
}

if (stat($dir . "/config.pl")->mtime < stat($pyfile)->mtime) {
  print STDERR "Warning: config.pl is older than config.py!\\n";
}

{
"""
print ’dbname => "%s",’ %(config.dbname)
print ’dbtype => "%s",’ %(config.dbtype)
print ’dbpass => "%s",’ %(config.dbpass)
print ’dbuser => "%s",’ %(config.dbuser)
print ’qdb_ports => %s,’ %(config.qdb_ports)
print ’qdb_host => "%s",’ %(config.qdb_host)
print ’align => "%s",’ %(config.align)
print ’al_jar => "%s",’ %(config.al_jar)
print ’tags => "%s",’ %(config.tags)
print ’databaseapi => "%s",’ %(config.databaseapi)
print ’tagger_de => "%s",’ %(config.tagger_de)
print ’tagger_en => "%s",’ %(config.tagger_en)
print ’wbsize => "%s",’ %(config.wbsize)
print ’wbids => "%s",’ %(config.wbids)
print ’goodcorpora => "%s",’ %(config.goodcorpora)
print ’#wbdateien => %s’ %(config. wbdateien)
print ’}’
```

Anhang L

Tagging – Code

L.1 tree-tagger-Skripte

L.1.1 tree-tagger-english

```
#!/bin/sh

# Set these paths appropriately
#HOME=$HOME/koks
[ -z $TAGGERHOME ] && TAGGERHOME=/home/koks
BIN=$TAGGERHOME/tagger/bin
CMD=$TAGGERHOME/tagger/cmd
LIB=$TAGGERHOME/tagger/lib
KOKSBIN=‘dirname "$0"‘

TOKENIZER=${BIN}/separate-punctuation
CORRECTION=${KOKSBIN}/metatag-correction
TAGGER=${BIN}/tree-tagger
ABBR_LIST=${LIB}/english-abbreviations
PARFILE=${LIB}/english.par
SENT_REC=${KOKSBIN}/punkt-tagger-english
SENT_REC_FIX=${KOKSBIN}/punkt.py          # Bug-Fix 21. Juni 2001

# put all on one line
cat $* |
# do tokenization
$TOKENIZER +1 +s +l $ABBR_LIST |
# separate clitics from preceding words
sed -e "s/’s"’$’"/ ’s/g" \
-e "s/s’"’$’"/ ’/g" \
-e "s/n’t"’$’"/ n’t/g" \
-e "s/’re"’$’"/ ’re/g" \
-e "s/’ve"’$’"/ ’ve/g" \
-e "s/’d"’$’"/ ’d/g" \
-e "s/’m"’$’"/ ’m/g" \
-e "s/’em"’$’"/ ’em/g" \
-e "s/’ll"’$’"/ ’ll/g" \
-e ’/^$/d’ |
tr ’ ’ ’\n’ |
# correct metatags
$CORRECTION |
# remove empty lines
grep -v ’^$’ |
# tagging
$TAGGER $PARFILE -token -lemma -sgml |
#$TAGGER $PARFILE -token -lemma -sgml
# recognize sentence endings
```

```
$SENT_REC |
$SENT_REC_FIX
```

L.1.2 tree-tagger-german

```
#!/bin/sh

# Set these paths appropriately
#HOME=$HOME/koks
[ -z $TAGGERHOME ] && TAGGERHOME=/home/koks

BIN=$TAGGERHOME/tagger/bin
CMD=$TAGGERHOME/tagger/cmd
LIB=$TAGGERHOME/tagger/lib
KOKSBIN=$TAGGERHOME/bin

TOKENIZER=${BIN}/separate-punctuation
CORRECTION=${KOKSBIN}/metatag-correction
GERMANIZER="perl -I${KOKSBIN} ${KOKSBIN}/umlaut.pl"
TAGGER=${BIN}/tree-tagger
ABBR_LIST=${LIB}/german-abbreviations
PARFILE=${LIB}/german-big.par
#PARFILE=${LIB}/german-small.par
FILTER=${CMD}/filter-german-tags
SENT_REC=${KOKSBIN}/punkt-tagger-german
SENT_REC_FIX=${KOKSBIN}/punkt.py          # Bug-Fix 21. Juni 2001
CLEANUP=${KOKSBIN}/cleanup.rb

# put all on one line
cat $* |
# do tokenization
$TOKENIZER +1 +s +l $ABBR_LIST |
# correct metatags
$CORRECTION |
# remove empty lines
grep -v '^$' |
# add Umlaut where necessary
$GERMANIZER |
# tagging
$TAGGER $PARFILE -token -lemma -sgml |
# error correction
$FILTER |
# recognize sentence endings
$SENT_REC |
$SENT_REC_FIX
```

L.2 Sonderzeichenrekonstruktion

```
#!/usr/bin/perl -w

use strict;
use DBI;

# $Header: /opt/cvs/bin/umlaut.pl,v 1.12 2001/08/03 12:03:41 koks-cvs Exp $
#
# umlaut.pl
#
# Output vom Tokenizer lesen und dabei versuchen, "ehemalige" Umlaute
# zu rekonstruieren.
```

```
# Benutzt Datenbank und ein bisschen Linguistik...

# lese Konfigurationsdatei ein
# config.pl wird mit py2pl aus config.py erzeugt :-)
my $config = require("config.pl");

my $user=$config->{dbuser};
my $passwd=$config->{dbpass};
my $dbname="DBI:mysql:dbname=" . $config->{dbname};

my $dbh = DBI->connect($dbname, $user, $passwd) || die("No database connection!");

while(my $word=<STDIN>){
  my $result = ersetze($word);
  print "$result\n";
}

$dbh->disconnect();

sub ersetze{
  my $word = $_[0];
  chomp $word;
  my $output = $word;
  my $known_in_db = 0;

  # unbekanntes Wort mit Umlaut/ß-Kandidaten?
  if (($word =~ /([aou]e|ss)/i)){
    # aufsplitten in Teile mit nur einem ([aou]e|ss)
    $word =~ s/(([aou]e|ss))/$1\t/ig;
    #$word =~ s/(([^e][aou]e|ss))/$1\t/ig;
    my @word = split /\t/, $word;
    my $length = @word;
    my $last = pop @word;

    # letztes Element ohne ([aou]e|ss)? -> mit vorletztem vereinen
    if (($last !~ /([aou]e|ss)/i) && ($length > 1)){
      push @word, (pop @word) . $last;
      $length--;
    }
    else{
      push @word, $last;
    }

    # alle Kandidaten generieren
    my @candidates = genAll(@word);
    my $cand_len = @candidates;

#    for(my $j = $cand_len-1; $j >= 0; $j--){
#      if (defined(dblookup($candidates[$j]))){
#        $output = $candidates[$j];
#      }
#    }

    foreach (@candidates){
      if (defined(dblookup($_))){
        $known_in_db = 1;
        $output = $_;
        last;
      }
    }
  }
```

```
  # für den Fall, daß das Wort unbekannt ist: alles ersetzen (außer ss)
  if($known_in_db == 0){
    $output =~ s/Ae/Ä/g;
    $output =~ s/Oe/Ö/g;
    $output =~ s/Ue/Ü/g;
    $output =~ s/ae/ä/g;
    $output =~ s/oe/ö/g;
    #$output =~ s/ue/ü/g;
    $output =~ s/([^aeqQ])ue/$1ü/g; # 'erneuern' != 'erneürn', 'Erbauer' != 'Erbaür', 'Quelle' != 'Qülle'
    my $rest = $output;
    my $anfang = "";
    # ss nach Diphtong zu ß außer an Kompositumsgrenze
    while($rest =~ /(au|ei|aeu|äu|eu)ss/){
      my @split = split //, $rest;
      while (($anfang !~ /s$/) || ($split[0] ne "s")){
        $anfang .= shift @split;
      }
      $rest =~ s/$anfang(.*)$/$1/g;
      # Kompositumsgrenze
      $rest = ersetze($rest);
      if(!defined(dblookup($rest))){
        if ($rest !~ /^s[cpt]/){
          $anfang =~ s/(.*)s$/$1ß/g;
          $rest =~ s/s(.*)$/$1/g;
        }
      }
    }
    $output = $anfang . $rest;
  }

  # Rückgabe
  return $output;
}

sub genAll{
  my @parts = @_;
  my @result = ();
  my $len = @parts;
  my $first = shift @parts;
  my $first2 = $first;
  my $subs = ($first =~ s/ae/ä/);
  $subs = ($first =~ s/Ae/Ä/) if !$subs;
  $subs = ($first =~ s/oe/ö/) if !$subs;
  $subs = ($first =~ s/Oe/Ö/) if !$subs;
  $subs = ($first =~ s/ue/ü/) if !$subs;
  $subs = ($first =~ s/Ue/Ü/) if !$subs;
  $subs = ($first =~ s/ss/ß/) if !$subs;
  if($len == 1){
    push @result, $first2;
    push @result, $first;
  }
  else{
    my @rest = genAll(@parts);
    foreach (@rest){
      push @result, $first2 . $_;
      push @result, $first . $_;
    }
  }
  return @result;
}
```

```
sub dblookup{
    my $text = $_[0];
    $_ = $text;
    s/'/''/;
    my $sth = $dbh->prepare("select t.name from tokens t, grundformen g where t.name = '" . $_ . "' and g.name <
    $sth->execute();
    my $result = ($sth->fetchrow())[0];
    $sth->finish();
    return $result;
}
```

L.3 Korrektur der Metatags

```
#! /usr/bin/perl -w


while($metatag = <STDIN>){
  $dotted = 0;
  chomp $metatag;
  if($metatag =~ /^([^\.]+)(\.\.+)$/){
    print "$1\n";
    $metatag = $2;
  }
  if($metatag =~ /^\.\.+$/){
    $dotted = 1;
    while(   (defined($next=<STDIN>))
          && ($next =~ /^\.+$/))
    {
       $metatag .= $next;
       chomp $metatag;
    }
    print "$metatag\n$next";
  }
  if(!$dotted){
    while ($metatag =~ /^<[^>]*$/){
      if(defined($next = <STDIN>)){
        $metatag = $metatag . " $next";
        chomp $metatag;
      }
    }
    print "$metatag\n";
  }
}
```

L.4 Satzendenerkennung

L.4.1 punkt-tagger-english

```
#! /usr/bin/perl -w


# ######################################################################
# Skript, das in der Ausgabe des englischen Tree-Taggers erkennt,     #
# ob es sich bei einem als "satzbeendendes Interpunktionzeichen"      #
# getaggten Zeichen tatsaechlich um einen solchen handelt;            #
# erkennt auch Punkte nach Ordinalzahlen oder Abkuerzungen als solche #
#                                                                     #
```

```
# Dieses Skript liegt auch fuer den deutschen Tree-Tagger vor         #
#                                                                     #
# liest von STDIN, schreibt nach STDOUT                               #
#                                                                     #
# ####################################################################

my %tags;
$tags{'punkt'} = "SENT";
$tags{'unknown'} = "<unknown>";
$tags{'satz'} = "<SATZ>";
$tags{'segment'} = "<segmentgrenze>";
$tags{'absatz'} = "<ABSATZ>";
$tags{'ende'} = "<DOKENDE>";

# Initialisierung

defined($zwei=<STDIN>) || exit 0;
#while($zwei =~ /^\s*</){
while($zwei !~ /^\s*[\w_äöüÄÖÜß\.]/){
  print "$tags{'satz'}\n$tags{'segment'}\n" if($zwei eq "$tags{'absatz'}");
  print $zwei;
  defined($zwei = <STDIN>) || exit 0;
}

print $zwei;                        # erste Zeile ausgeben, die wird mangels
                                    # Kontext nicht ueberprueft

defined($drei=<STDIN>) || exit 0;
# while($drei =~ /^\s*</){
while($drei !~ /^\s*[\w_äöüÄÖÜß\.]/){
  print "$tags{'satz'}\n$tags{'segment'}\n" if($drei eq "$tags{'absatz'}");
  print $drei;
  defined($drei = <STDIN>) || exit 0;
}

#######################################################################

# los geht's

while($neueZeile=<STDIN>) {          # neue Zeile einlesen

  # Schluss machen, wenn <dokende>
  $trash = $neueZeile;
  chomp($trash);
  if (lc($trash) eq lc($tags{'ende'})) {
    last;
  }

  $unveraendert  =  1;
  $commentBuffer = "";

  $eins = $zwei;                    # Zeilenfenster aktualisieren
  $zwei = $drei;
  $drei = $neueZeile;

  # Kommentarzeilen ausfiltern

  #while($drei =~ /^\s*</){
  while($drei !~ /^\s*[\w_äöüÄÖÜß\.!\?]/){
    $commentBuffer = $commentBuffer . "$tags{'satz'}\n$tags{'segment'}\n" if($drei eq "$tags{'absatz'}");
    $commentBuffer = $commentBuffer . $drei;
```

```
  if (!(defined($drei = <STDIN>))){
    print $commentBuffer;
    exit 0;
  }
}

chomp ($eins, $zwei, $drei);

@eins = split /\t/, $eins;        # Zugriff auf Wort, Tag, Lemma schaffen
@zwei = split /\t/, $zwei;
@drei = split /\t/, $drei;

if($zwei[1] eq $tags{'punkt'}){        # wenn mittlere Zeile als Punkt getagt
  # Wort nach Punkt groß, Lemma klein -> Satzende

  if (    ($drei[2] !~ /^</)               # Lemma != <unknown>
       && (lcfirst($drei[0]) ne $drei[0])   # Wort groß
       && (lcfirst($drei[2]) eq $drei[2])   # Lemma klein
     )
  {
    print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
    next;
  }

  # Zahl vor Punkt, danach Monatsname oder anderes typisches Wort
  # -> Ordinalzahl

  if ($eins[0] =~ /[0-9]+/ && $monatsname{$drei[2]}){
    print "$zwei[0]\tORD-P\t$zwei[2]\n$commentBuffer";
    next;
  }

  # Wort vor Punkt ohne Vokal -> Abkuerzung

  if ($eins[0] !~ /[AEIOUÄÖÜYaeiouäöüy]/){
    print "$zwei[0]\tABK-P\t$zwei[2]\n$commentBuffer";
    next;
  }

  # Wort vor Punkt hat unzulässigen absoluten Endrand -> Abkürzung
  # zulaessige engl. Endraender?

  # <ABSATZ>-Markierung bedeutet auch Satzgrenze

  if($zwei =~ /^<ABSATZ>$/i){
    print "$tags{'satz'}\n$tags{'segment'}\n$tags{'absatz'}\n$commentBuffer";
    next;
  }

  # Default: Satzgrenze

  if ($unveraendert){
    print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
    next;
  }
}

# Auch wenn kein abgerueckter Punkt:
# Wort nach Punkt groß, Lemma klein -> Abkuerzung am Satzende

if (($zwei[0] =~ /.+\.$/)
```

```
      && ($drei[2] !~ /^</)
      && (lcfirst($drei[0]) ne $drei[0])
      && (lcfirst($drei[2]) eq $drei[2]))
  {
    print "$zwei[0]\t$zwei[1]\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
    $unveraendert = 0;
  }

  # Ansonsten Zeile ausgeben

  print "$zwei\n$commentBuffer" if ($unveraendert);
}

# Ganz zum Schluß: Letzte Zeile ausgeben

if($drei[1] eq $tags{'punkt'}){
  print "$drei[0]\tSATZ-P\t$drei[2]\n$tags{'satz'}\n$tags{'segment'}\n";
}
else{
  print "$drei\n$tags{'satz'}\n$tags{'segment'}\n";
}
print "$tags{'ende'}\n";

##########################################################################

# ein paar nuetzliche Hashes

%monatsname = ( 'january' => 1,
                'february' => 1,
                'march' => 1,
                'april' => 1,
                'may' => 1,
                'june' => 1,
                'july' => 1,
                'august' => 1,
                'september' => 1,
                'october' => 1,
                'november' => 1,
                'december' => 1,
                'place' => 1,
                'price' => 1,
              );
```

L.4.2 punkt-tagger-german

```
#!/usr/bin/perl -w

# ########################################################################
# Skript, das in der Ausgabe des deutschen Tree-Taggers erkennt,      #
# ob es sich bei einem als "satzbeendendes Interpunktionzeichen"      #
# getaggten Zeichen tatsaechlich um einen solchen handelt;            #
# erkennt auch Punkte nach Ordinalzahlen oder Abkuerzungen als solche #
#                                                                     #
# Dieses Skript liegt auch fuer den englischen Tree-Tagger vor        #
#                                                                     #
# liest von STDIN, schreibt nach STDOUT                               #
#                                                                     #
# ########################################################################

my %tags;
```

```
$tags{'punkt'} = "\$.";
$tags{'unknown'} = "<unknown>";
$tags{'satz'} = "<SATZ>";
$tags{'segment'} = "<segmentgrenze>";
$tags{'absatz'} = "<ABSATZ>";
$tags{'ende'} = "<DOKENDE>";

# Initialisierung

defined($zwei=<STDIN>) || exit 0;
#while($zwei =~ /^\s*</){
while($zwei !~ /^\s*[\w_äöüÄÖÜß\.]/){
  print "$tags{'satz'}\n$tags{'segment'}\n" if ($zwei eq $tags{'absatz'});
  print $zwei;
  defined($zwei = <STDIN>) || exit 0;
}

print $zwei;                        # erste Zeile ausgeben, wird mangels
                                    # Kontext nicht ueberprueft

defined($drei=<STDIN>) || exit 0;
#while($drei =~ /^\s*</){
while($drei !~ /^\s*[\w_äöüÄÖÜß\.]/){
  print "$tags{'satz'}\n$tags{'segment'}\n" if ($drei eq $tags{'absatz'});
  print $drei;
  defined($drei = <STDIN>) || exit 0;
}

###############################################################################

# los geht's
while($neueZeile=<STDIN>){          # neue Zeile einlesen

  # Schluss bei <dokende>
  $trash = $neueZeile;
  chomp($trash);
  if (lc($trash) eq lc($tags{'ende'})) {
    #print STDERR "Hab Dich: $trash\n";
    last;
  }
  #print STDERR "Normal: $trash\n";

  # Zeilen mit Müll gleich wegwerfen
  #@trash = split /\t/, $trash;
  #if (($trash[2] eq $UNKNOWN) && (length($trash[0]) > 1)) {
    ##print STDERR "Tata '$trash[0]' -->";
    #if ($trash[0] !~ /[a-zA-ZäöüÄÖÜß0-9]+/) {
      ##print STDERR "[-]\n";
      #next;
    #}
    ##print STDERR "[+]\n";
  #}

  $unveraendert  =  1;
  $commentBuffer = "";

  $eins = $zwei;                    # Zeilenfenster aktualisieren
  $zwei = $drei;
  $drei = $neueZeile;

  # Kommentarzeilen ausfiltern
```

```
# while($drei =~ /^\s*</)
while($drei !~ /^\s*[\w_äöüÄÖÜß\.!\?]/){
  $commentBuffer = $commentBuffer . "$tags{'satz'}\n$tags{'segment'}\n" if($drei eq $tags{'absatz'});
  $commentBuffer = $commentBuffer . $drei;
  if (!(defined($drei = <STDIN>))){
    print $commentBuffer;
    exit 0;
  }
}

chomp ($eins, $zwei, $drei);

@eins = split /\t/, $eins;        # Zugriff auf Wort, Tag, Lemma schaffen
@zwei = split /\t/, $zwei;
@drei = split /\t/, $drei;

if($zwei[1] eq $tags{'punkt'}){        # wenn mittlere Zeile als Punkt getagt
  if($zwei[0] eq "."){

    # Wort nach Punkt groß, Lemma klein -> Satzende

    if (    ($drei[2] !~ /^</)               # Lemma != <unknown>
         && (lcfirst($drei[0]) ne $drei[0])   # Wort groß
         && (lcfirst($drei[2]) eq $drei[2])   # Lemma klein
       )
    {
      print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
      next;
    }

    # Zahl vor Punkt, danach Monatsname oder anderes typisches Wort
    # -> Ordinalzahl

    if ($eins[0] =~ /[0-9]+/ && $monatsname{$drei[2]}){
      print "$zwei[0]\tORD-P\t$zwei[2]\n$commentBuffer";
      next;
    }

    # Wort vor Punkt ohne Vokal -> Abkuerzung

    if(   ($eins[0] !~ /[AEIOUÄÖÜYaeiouäöüy]/)
        && ($eins[0] =~ /[\wß]/))
    {
      print "$zwei[0]\tABK-P\t$zwei[2]\n$commentBuffer";
      next;
    }

    # Wort vor Punkt hat unzulässigen absoluten Endrand -> Abkürzung
#
#    $endrand = $eins[0];
#    $endrand =~ s/^.*([^aeiouäöüy]*)$/$1/g;
#    if (! $endrand{$endrand}){
#      print "$zwei[0]\tABK-P\t$zwei[2]\n";
#      print $commentBuffer;
#      $unveraendert = 0;
#      next;
#    }
#

    # <ABSATZ>-Markierung bedeutet auch Satzgrenze
```

```
      if($zwei =~ /^<ABSATZ>$/i){
        print "$tags{'satz'}\n$tags{'segment'}\n$tags{'absatz'}\n$commentBuffer";
        next;
      }

      # Default: Satzgrenze

      if ($unveraendert){
        print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
        next;
      }
    }
    elsif ($zwei[0] eq "!" || $zwei[0] eq "?") {
      # Nach '?' und '!' immer Satzende
      print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
      next;
    }

#    if($zwei[0] eq ":"){
#      # Wort nach Punkt groß, Lemma klein -> Satzende
#
#      if (    ($drei[2] !~ /^</)               # Lemma != <unknown>
#           && (lcfirst($drei[0]) ne $drei[0])   # Wort groß
#           && (lcfirst($drei[2]) eq $drei[2])   # Lemma klein
#         )
#      {
#        print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
#        next;
#      }
#
#      # <ABSATZ>-Markierung bedeutet auch Satzgrenze
#
#      if($zwei =~ /^<ABSATZ>$/i){
#        print "$tags{'satz'}\n$tags{'segment'}\n$tags{'absatz'}\n$commentBuffer";
#        next;
#      }
#    }
  }

  # Auch wenn kein abgerueckter Punkt:
  # Wort nach Punkt groß, Lemma klein -> Abkuerzung am Satzende

  if (($zwei[0] =~ /.+\.$/)
      && ($drei[2] !~ /^</)
      && (lcfirst($drei[0]) ne $drei[0])
      && (lcfirst($drei[2]) eq $drei[2]))
  {
    print "$zwei[0]\t$zwei[1]\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
    $unveraendert = 0;
  }

  # nur Punkte (z.B. "....")

  #if ($zwei[0] =~ /^\.\.+$/){
  #  print "$zwei[0]\t\$(\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
  #  $unveraendert = 0;
  #}

  # Ansonsten Zeile ausgeben
```

```
  print "$zwei\n$commentBuffer" if ($unveraendert);
}

# Ganz zum Schluß: Letzte Zeile ausgeben

if($drei[1] eq $tags{'punkt'}){
  print "$drei[0]\tSATZ-P\t$drei[2]\n$tags{'satz'}\n$tags{'segment'}\n";
}
else{
  print "$drei\n$tags{'satz'}\n$tags{'segment'}\n";
}
print "$tags{'ende'}\n";

###############################################################################

# ein paar nuetzliche Hashes

%monatsname = ( 'Januar' => 1,
                'Februar' => 1,
                'März' => 1,
                'April' => 1,
                'Mai' => 1,
                'Juni' => 1,
                'Juli' => 1,
                'August' => 1,
                'September' => 1,
                'Oktober' => 1,
                'November' => 1,
                'Dezember' => 1,
                'Platz' => 1,
                'Preis' => 1,
                'Versuch' => 1,
                'Anlauf' => 1
              );

#%endrand = ('' => 1,
#            'b' => 1, 'bb' => 1, 'bbs' => 1, 'bbst' => 1, 'bs' => 1,
#                'bsch' => 1, 'bschs' => 1, 'bschst' => 1, 'bscht' => 1,
#                'bst' => 1, 'bsts' => 1,
#            'ch' => 1, 'chs' => 1, 'chst' => 1, 'ck' => 1, 'cks' => 1,
#                'ckst' => 1,
#            'd' => 1, 'dd' => 1, 'ds' => 1, 'dst' => 1,
#            'f' => 1, 'ff' => 1, 'ffs' => 1, 'ffst' => 1, 'fs' => 1, 'fst' => 1,
#            'g' => 1, 'gg' => 1, 'ggs' => 1, 'ggst' => 1, 'gs' => 1, 'gst' => 1,
#            'h' => 1,
#            'k' => 1, 'ks' => 1, 'kst' => 1,
#            'l' => 1, 'lb' => 1, 'lbs' => 1, 'lbst' => 1, 'lbt' => 1, 'ld' => 1,
#                'lds' => 1, 'ldst' => 1, 'lf' => 1, 'lfs' => 1, 'lfst' => 1,
#                'lft' => 1, 'lg' => 1, 'lgs' => 1, 'lgst' => 1, 'lgt' => 1,
#                'lk' => 1, 'lks' => 1, 'lkst' => 1, 'lkt' => 1, 'll' => 1,
#                'lls' => 1, 'llst' => 1, 'lm' => 1, 'lms' => 1, 'lmst' => 1,
#                'lmt' => 1, 'ln' => 1, 'lns' => 1, 'lnst' => 1, 'lnt' => 1,
#                'lp' => 1, 'lps' => 1, 'lpst' => 1, 'lpt' => 1, 'ls' => 1,
#                'lsch' => 1, 'lschs' => 1, 'lschst' => 1, 'lscht' => 1,
#                'lst' => 1, 'lt' => 1, 'lts' => 1, 'ltst' => 1, 'lz' => 1,
#                'lzt' => 1,
#            'm' => 1, 'mb' => 1, 'mbs' => 1, 'mbst' => 1, 'mbt' => 1, 'md' => 1,
#                'mds' => 1, 'mdst' => 1, 'mf' => 1, 'mfs' => 1, 'mfst' => 1,
#                'mft' => 1, 'mm' => 1, 'mms' => 1, 'mmst' => 1, 'mp' => 1,
#                'mpf' => 1, 'mpfs' => 1, 'mpftmpfst' => 1, 'mps' => 1,
#                'mpst' => 1, 'mpt' => 1, 'ms' => 1, 'mst' => 1, 'mt' => 1,
```

```
#               'mts' => 1, 'mtst' => 1, 'mz' => 1, 'mzt' => 1,
#           'n' => 1, 'nd' => 1, 'nds' => 1, 'ndst' => 1, 'ndt' => 1, 'nf' => 1,
#               'nfs' => 1, 'nfst' => 1, 'nft' => 1, 'ng' => 1, 'ngs' => 1,
#               'ngst' => 1, 'ngt' => 1, 'nk' => 1, 'nks' => 1, 'nkst' => 1,
#               'nkt' => 1, 'nn' => 1, 'nns' => 1, 'nnst' => 1, 'ns' => 1,
#               'nsch' => 1, 'nschs' => 1, 'nschst' => 1, 'nscht' => 1,
#               'nst' => 1, 'nt' => 1, 'nts' => 1, 'ntst' => 1, 'nz' => 1,
#               'nzt' => 1,
#           'p' => 1, 'pf' => 1, 'pfs' => 1, 'ppst' => 1, 'ps' => 1, 'pt' => 1,
#           'r' => 1, 'rb' => 1, 'rbs' => 1, 'rbst' => 1, 'rbsts' => 1,
#               'rbt' => 1, 'rd' => 1, 'rds' => 1, 'rdst' => 1, 'rf' => 1,
#               'rfs' => 1, 'rfst' => 1, 'rft' => 1, 'rg' => 1, 'rgs' => 1,
#               'rgst' => 1, 'rgt' => 1, 'rk' => 1, 'rks' => 1, 'rkst' => 1,
#               'rkt' => 1, 'rl' => 1, 'rls' => 1, 'rm' => 1, 'rms' => 1,
#               'rmst' => 1, 'rmt' => 1, 'rn' => 1, 'rns' => 1, 'rnst' => 1,
#               'rnt' => 1, 'rp' => 1, 'rps' => 1, 'rpst' => 1, 'rpt' => 1,
#               'rrst' => 1, 'rs' => 1, 'rsch' => 1, 'rschs' => 1,
#               'rschst' => 1, 'rscht' => 1, 'rst' => 1, 'rt' => 1, 'rts' => 1,
#               'rtst' => 1, 'rz' => 1, 'rzt' => 1,
#           's' => 1, 'sch' => 1, 'schs' => 1, 'schst' => 1, 'scht' => 1,
#               'sk' => 1, 'ss' => 1, 'sst' => 1, 'st' => 1, 'sts' => 1,
#           't' => 1, 'ts' => 1, 'tt' => 1, 'tts' => 1, 'tz' => 1, 'tzt' => 1,
#           'v' => 1,
#           'x' => 1, 'xt' => 1,
#           'z' => 1, 'zt' => 1
#);
#
```

Anhang M

Alignment – Code

M.1 Region Paket

Die Quellen des Python-Pakets `Region` befinden sich im CVS Baum im Verzeichnis `align/Region`.

M.1.1 __init__.py

```
#!/usr/bin/python

# Initialisierung des Pakets "Region"

__all__ = [
           "absatz.py",
           "atom.py",
           "dokument.py",
           "korpus.py",
           "region.py",
           "satz.py",
           "segment.py"
]
```

M.1.2 absatz.py

```
#!/usr/bin/python

# Modul Region.absatz

# KOKS Studienprojekt 2000 - 2001
# Korpusbasierte Kollokationensuche

import alignment
import region
import segment
import string

class Absatz(region.Regionen): #{

    def __repr__(self):
        rtext = self.getText()
        rtext = string.replace(rtext, "\n", " ")
        rtext = string.replace(rtext, "\t", " ")
        if len(rtext) > 64:
            rtext = rtext[:60] + "..."
        return "<Absatz: %s>" %rtext

    def getText(self):
        return region.Regionen.getText(self) + "\n"
```

```
25
26      def getDelimiter(self):
27          return "<ABSATZ>\n"
28
29      delimiter = "<ABSATZ>\n"
30
31      def quelleninfo(self):
32          return "Absatz %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)
33
34  #}
35
36  class AbsatzPaar(Absatz, alignment.AlignedRegion): #{
37
38      def __init__(self, abs1, abs2, qs, zs):
39          alignment.AlignedRegion.__init__(self, abs1, abs2, qs, zs)
40          Absatz.__init__(self, abs1)
41
42      def alignedRegion(self):    # liefert den alignten Absatz
43          return AbsatzPaar(self.reg2, self.reg1, self.lang2, self.lang1)
44
45      def getItemAtIndex(self, index):
46          segment1 =  self.reg1[index]
47          try:
48              segment2 =  self.reg2[index]
49          except IndexError:
50              return self.reg1[index]
51          retval = segment.SegmentPaar(segment1, segment2, self.lang1, self.lang2)
52          retval.setQuelleninfo(index, self.quelleninfo())
53          return retval
54
55  #}
56
57  class AbsatzVonZeilenliste(Absatz): #{
58
59    def __init__(self, zeilenliste):
60      segmentliste = []
61      pos = 0
62      start = 0
63      inhaltGelesen = 0
64      for zeile in zeilenliste: #{
65        if zeile[0:15] == "<segmentgrenze>": #{
66          segmentliste.append(
67                          segment.SegmentVonZeilenliste(zeilenliste[start:pos]))
68          start = pos + 1
69          inhaltGelesen = 0
70        #}
71        else: #{
72          inhaltGelesen = 1
73        #}
74        pos = pos + 1
75      #}
76      if inhaltGelesen: #{
77        segmentliste.append(
78                          segment.SegmentVonZeilenliste(zeilenliste[start:pos]))
79      #}
80      Absatz.__init__(self, segmentliste)
81  #}
82
83  if __name__ == '__main__':
84      import atom
85      a = "Äpfel\tNN\tApfel\n"
```

```
    b = "Birnen\tNN\tBirne\n"
    c = "<segmentgrenze>\n"
    d = "Cashewnüsse\tNN\tCashewnuß\n"
    e = "Datteln\tNN\tDattel\n"
    f = "<segmentgrenze>\n"
    g = AbsatzVonZeilenliste([a,b,c,d,e,f,a,b,c])
    print "Absatz [a,b,...,a,b,...]:", g, `g`
    r = g.getRegionenMit(segment.Segment)
    print "Regionen mit Segment:", r, `r`, len(r)
    for i in range(0, len(r)):
        print ("["+`i`+"]: " + `r[i]`)
    r = g.getRegionenMit(atom.Atom)
    print r, `r`, len(r)
    for i in range(0, len(r)):
        print ("["+`i`+"]: " + `r[i]`)
```

M.1.3 alignment.py

```
#!/usr/bin/python

# Modul Region.alignment

import sys
import types
import region

class AlignedRegion:

    def __init__(self, reg1, reg2, lang1, lang2): # i.d.R. nicht verwendet
        self.reg1 = reg1
        self.reg2 = reg2
        self.lang1 = lang1
        self.lang2 = lang2

    def alignedRegion(self):     # ueberschreiben
        return None              # liefert die Region, die zu self alignt ist

    def getQuellsprache(self):
        return "undefiniert"

    def getZielsprache(self):
        return "undefiniert"

```

M.1.4 atom.py

```
#!/usr/bin/python

import region
import string
import sys

class Atom(region.Region): #{

  def __repr__(self):
    return "<Atom" + `( self.feld["Wort"],
                        self.feld["Tag"],
                        self.feld["Lemma"] )` + ">"

  def __init__(self, wort, tag, lemma):
    region.Region.__init__(self)
```

```
    self.feld = {}
    self.feld["Wort"]  = wort
    self.feld["Tag"]   = tag
    self.feld["Lemma"] = lemma

  def getFeldinhalt(self, feldname):
    return [self.feld[feldname]]

  def getDelimiter(self):
    return ""                     # muss Wahrheitswert falsch haben

  delimiter = ""

#}

class AtomVonZeile(Atom): #{

  def __init__(self, zeile):
    spalte = string.split(zeile, "\t")    # am Tabulator trennen
    if len(spalte) != 3:
      self.ok = 0
      return
    wort   = spalte[0]
    tag    = spalte[1]
    lemma  = spalte[2]
    lemma  = string.rstrip(lemma)         # Newline oder anderen WS tilgen
    Atom.__init__(self, wort, tag, lemma)
    self.ok = 1

  def isOK(self):
    return self.ok
#}

if 0: #{
  a = AtomVonZeile("Äpfel\tNN\tApfel\n")
  print "Atom a:", a, `a`
  b = AtomVonZeile("Birnen\tNN\tBirne\n")
  print "Atom b:", b, `b`
#}
```

M.1.5 dokument.py

```
#!/usr/bin/python

# Modul Region/dokument

# KOKS Studienprojekt 2000 - 2001
# Korpusbasierte Kollokationensuche

import absatz
import alignment
import region
import string
import sys
import time

class Dokument(region.Regionen): #{

    def __repr__(self):
        return "<Dokument: " + `self.subregionen`+ ">"

```

```
    def getText(self):
        return string.replace(region.Regionen.getText(self), "\n ", "\n")

    def getDelimiter(self):
        return "<DOKENDE>\n"

    delimiter = "<DOKENDE>\n"

    def quelleninfo(self):
        return "Dokument %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)

#}

class DokumentVonStream(Dokument): #{

    def __repr__(self):
        return "<DokumentVonStream>"

    def __init__(self, f): #{
        zeile = f.readline()
        absatzliste = []
        zeilenliste = []
        ignorieren=0
        inhaltGelesen = 0
        while zeile != "": #{
            if zeile[0:8] == "<ABSATZ>":
                absatzliste.append(absatz.AbsatzVonZeilenliste(zeilenliste))
                zeilenliste = []
                inhaltGelesen = 0
                ignorieren=0                    # <KOKS> komplett zuruecksetzen
            elif zeile[0:6] == "<KOKS>":
                ignorieren = ignorieren + 1
            elif zeile[0:7] == "</KOKS>":
                ignorieren = ignorieren - 1
            elif zeile[0:9] == "<DOKENDE>":    # Dokument.delimiter
                break
            elif zeile[0:9] == "<dokende>":    # kommt in den de-news vor
                break
            elif zeile[0:12] == "<KorpusEnde>":  # zur Sicherheit
                break
            elif not ignorieren:
                zeilenliste.append(zeile)
                inhaltGelesen = 1
            zeile = f.readline()
        #}
        if inhaltGelesen:
            absatzliste.append(absatz.AbsatzVonZeilenliste(zeilenliste))
        Dokument.__init__(self, absatzliste)
    #}

    def quelleninfo(self):
        return "Dokument von Eingabestrom"

#}

class DokumentVonDatei(DokumentVonStream): #{

    def __repr__(self):
        return "<Dokument: " + self.dateiname + ">"

    def __init__(self, dateiname):
```

```
81          self.dateiname = dateiname
82          f = open(dateiname, "r")
83          DokumentVonStream.__init__(self, f)
84
85      def quelleninfo(self):
86          return "Dokument-Datei %s" %self.dateiname
87
88  #}
89
90  class DokumentPaar(Dokument, alignment.AlignedRegion): #{
91
92      def __init__(self, dok1, dok2, qs, zs):
93          alignment.AlignedRegion.__init__(self, dok1, dok2, qs, zs)
94          Dokument.__init__(self, dok1)
95
96      def alignedRegion(self):    # liefert das alignte Dokument
97          return DokumentPaar(self.reg2, self.reg1, self.lang2, self.lang1)
98
99      def getItemAtIndex(self, index):
100         absatz1 =  self.reg1[index]
101         try:
102             absatz2 =  self.reg2[index]
103         except IndexError:
104             sys.stderr.write("Warnung: %s hat verschieden viele Absaetze\n" %‘self‘)
105             return self.reg1[index]
106         retval = absatz.AbsatzPaar(absatz1, absatz2, self.lang1, self.lang2)
107         retval.setQuelleninfo(index, self.quelleninfo())
108         return retval
109
110     def quelleninfo(self):
111         info1 = self.reg1.quelleninfo()
112         info2 = self.reg2.quelleninfo()
113         return "Paar %s und %s" %(info1, info2)
114
115 #}
116
117 class DokumentPaarVonDateien(DokumentPaar): #{
118
119     def __init__(self, fnameqs, fnamezs, qs, zs):
120         dok1 = DokumentVonDatei(fnameqs)     # Quellsprache
121         dok2 = DokumentVonDatei(fnamezs)     # Zielsprache
122         DokumentPaar.__init__(self, dok1, dok2, qs, zs)
123
124     def quelleninfo(self):
125         info1 = self.reg1.quelleninfo()
126         info2 = self.reg2.quelleninfo()
127         if info1[-7:] == ".de.tag" and info2[-7:] == ".en.tag" \
128         and info1[:-7] == info2[:-7]:
129             return info1[:-7]
130         else:
131             return "Paar %s und %s" %(info1, info2)
132
133 #}
134
135 class DokumentPaarVonStreams(DokumentPaar): #{
136
137     def __init__(self, finqs, finzs, qs, zs):
138         dok1 = DokumentVonStream(finqs)     # Quellsprache
139         dok2 = DokumentVonStream(finzs)     # Zielsprache
140         DokumentPaar.__init__(self, dok1, dok2, qs, zs)
141
```

```
#}

class DokumentPaarVonRohtexte(DokumentPaar): #[

    def __init__(self, fin1, fin2, qs, zs, verbose = 0):
        absatzdic = {}
        absatzdic[qs] = fin1.readline()
        absatzdic[zs] = fin2.readline()
        if verbose:
            sys.stderr.write("tagging %s...\n" %‘absatzdic‘)
        getaggt = self.p_taggen(absatzdic)
        if verbose:
            for key in getaggt.keys():
                sys.stderr.write("result[%s]:\n" %key)
                getaggt[key].writeTagzeilen(sys.stderr.write)
        absatzliste1 = []
        absatzliste2 = []
        while getaggt:
            absatzliste1.append(getaggt[qs])
            absatzliste2.append(getaggt[zs])
            absatzdic[qs] = fin1.readline()
            absatzdic[zs] = fin2.readline()
            if verbose:
                sys.stderr.write("tagging %s...\n" %‘absatzdic‘)
            getaggt = self.p_taggen(absatzdic)
            if verbose:
                for key in getaggt.keys():
                    sys.stderr.write("result[%s]:\n" %key)
                    getaggt[key].writeTagzeilen(sys.stderr.write)
        dok1 = Dokument(absatzliste1)
        dok2 = Dokument(absatzliste2)
        DokumentPaar.__init__(self, dok1, dok2, qs, zs)

    def p_taggen(self, absatzdic):
        tagdic = {}
        for key in absatzdic.keys():
             einAbsatz = absatzdic[key]
             if not einAbsatz:
                 return {}
             getaggt = tagger.tagline(einAbsatz, key)
             tagdic[key] = getaggt
        return tagdic

#}

import tagger
```

M.1.6 korpus.py

```
#!/usr/bin/python

# Modul Region/korpora

# KOKS Studienprojekt 2000 - 2001
# Korpusbasierte Kollokationensuche

import alignment
import dokument
import region
import string

```

```
class Korpus(region.Regionen): #{
  def __repr__(self):
    return "<Korpus: " + ‘self.subregionen‘+ ">"

  def getText(self):
    list = []
    for i in region.Regionen.keys():
      list.append(i.getText())
    return string.join(list, "\n" + 72*"-" + "\n\n")

  def getDelimiter(self):
    return "<KorpusEnde>\n"

  delimiteri = "<KorpusEnde>\n"

  def quelleninfo(self):
      return "Korpus\n"

#}

class KorpusMitSprache(Korpus): #{

  def __init__(self, dokumentliste, sprache):
    self.sprache = sprache
    Korpus.__init__(self, dokumentliste)

  def p_setSprache(self, sprache):   # nur in dieser Klasse
    self.sprache = sprache

  def p_setDokumentliste(self, liste):     # nur in dieser Klasse
    self.subregionen = liste

#}

class KorpusVonDateiliste(Korpus): #{

  def __init__(self, dateiliste):
    liste = []
    for i in dateiliste:
      liste.append(dokument.DokumentVonDatei(i))
    Korpus.__init__(self, liste)
#}

class KorpusPaar(Korpus, alignment.AlignedRegion): #{

    def __init__(self, korp1, korp2, lang1, lang2):
        alignment.AlignedRegion.__init__(self, korp1, korp2, lang1, lang2)
        Korpus.__init__(self, korp1)      # Quellsprache

    def alignedRegion(self):    # liefert den alignten Korpus
        return KorpusPaar(self.reg2, self.reg1, self.lang2, self.lang1)

    def getItemAtIndex(self, index):
        dok1 =  self.reg1[index]
        try:
            dok2 =  self.reg2[index]
        except IndexError:
            return self.reg1[index]
        return dokument.DokumentPaar(dok1, dok2, self.lang1, self.lang2)

#}
```

M.1.7 region.py

```
#!/usr/bin/python

# abtrakte Basisklasse fuer alle Arten von Regionen: Atom, Segment,
# Absatz und Dokument.

import bisect
import string
import sys
import types

class Region: #{

    def __init__(self):
        self.setQuelleninfo(-1, "unbekannt")
        #self.cache = cache.DefaultCache(self)

    def __repr__(self):
        return "<abstrakte Region>"            # diese Methoden ueberschreiben

    def p_getFeldinhalt(self, feldname):
        if self.cache.has_key(feldname):
            return self.cache[feldname]
        retval = self.getFeldinhalt(feldname)
        self.cache[feldname] = retval
        return retval

    def getFeldinhalt(self, feldname):         # diese Methoden ueberschreiben
        return []

    def getWoerter(self):
        return self.getFeldinhalt("Wort")

    def getTags(self):
        return self.getFeldinhalt("Tag")

    def getLemmata(self):
        return self.getFeldinhalt("Lemma")

    def getText(self):
        return string.join(self.getWoerter())

    def __str__(self):
        return self.getText()

#    def getDelimiter(self):                    # Interface !!!
#        return "<Delimiter>\n"
#
#    delimiter = "<Delimiter>\n"

    def writeTagzeilen(self, fun):
         mm=map(None, self.getWoerter(),
                      self.getTags(),
                      self.getLemmata())
         for (w,t,l) in mm:
             apply(fun, ("%s\t%s\t%s\n" %(w,t,l),))

    def quelleninfo(self):
         return "Region %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)

```

```
60      def setQuelleninfo(self, index, von):
61           self.quelleninfo_index = index
62           self.quelleninfo_von = von
63
64  #}
65
66  class Regionen(Region): #{
67
68      def __repr__(self):                                 # diese Methoden ueberschreiben
69          return "<abstrakte Regionen>"
70
71      def __init__(self, regionenliste):
72          Region.__init__(self)
73          self.subregionen = regionenliste
74
75      def getFeldinhalt(self, feldname):
76          liste = []
77          for i in self.keys():
78              liste = liste + self[i].getFeldinhalt(feldname)
79          return liste
80
81      def getText(self):                                  # nicht von Region uebernommen,
82          liste = []                                      # damit Aenderungen an getText
83          for i in self.keys():                           # auch weitergereicht werden
84              liste.append(self[i].getText())
85          return string.join(liste)
86
87      def __len__(self):                                  # ueberschreiben, wenn getRegionenZu
88          return len(self.subregionen)                    # funktionieren soll
89
90      def has_key(self, key):
91          return 0 <= key < len(self)
92
93      def keys(self):
94          return range(0, len(self))
95
96      def getItemAtIndex(self, index):
97          retval = self.subregionen[index]
98          retval.setQuelleninfo(index, self.quelleninfo())
99          return retval
100
101     def index(self, obj):
102         for i in range(0, self.__len__()):
103             cmp = self[i]
104             if cmp is obj:
105                 return i
106         raise ValueError, "%s not in list!" %obj
107
108     def count(self, obj):
109         counter = 0
110         for i in range(0, self.__len__()):
111             cmp = self[i]
112             #print cmp, obj
113             if cmp is obj:
114                 counter = counter + 1
115         return counter
116
117     def __getitem__(self, key):
118         if type(key) == types.IntType:
119             return self.getItemAtIndex(key)
120         if type(key) == types.SliceType:
```

```
121             return RegionenSlice(self, key)
122         raise TypeError
123 
124     def __getslice__(self, a, b):        # Python 1.5 Kompatibilitaet
125         return RegionenSlice(self, Slice15(a, b))
126 
127     def getRegionenMit(self, klasse):
128         return RegionenMit(self, klasse)
129 
130     def writeTagzeilen(self, fun):
131 #         apply(fun, ("DEBUG: vor keys() %s" %self.keys(),))
132         for i in self.keys():                                  # auch weitergereicht werden
133             self[i].writeTagzeilen(fun)
134 #         apply(fun, ("DEBUG: nach schleife",))
135         try:
136             delimiter = self.getDelimiter()
137         except AttributeError:
138             delimiter = "<unbestimmter Regionentyp>\n"
139         apply(fun, ("%s" %delimiter,))
140 
141 #}
142 
143 class Slice15: #{                    # Hilfsklasse zur Python 1.5 Kompatibilitaet
144     def __init__(self, a, b):
145         self.start = a
146         self.stop    = b
147         self.step    = None
148 #}
149 
150 class RegionenMit(Regionen): #{
151 
152     def __repr__(self):
153         return "<Regionen mit " + self.klasse.__name__ + ":%s>" %self.p_repr2()
154 
155     def p_repr2(self):
156         return "\n\t"+‘(self.pos, self.subregionen)‘
157 
158     def __init__(self, regionen, klasse): #{
159         self.klasse = klasse
160         delimiter = klasse.delimiter                # workaround: delimiter benutzen
161         if len(regionen) == 0:
162             self.pos = [0]
163             Regionen.__init__(self, [])
164         #elif isinstance(regionen[0], klasse):
165         elif regionen[0].getDelimiter() == delimiter:
166             self.pos = [0, len(regionen)]
167             Regionen.__init__(self, [regionen])
168         else: #{
169             liste = []
170             start = 0
171             self.pos = [0]
172             import time
173             import sys
174             letzteAusgabe = 0
175             rname = regionen.__class__.__name__
176             kname = klasse.__name__
177             for i in regionen.keys(): #{
178                 subregionenMit = RegionenMit(regionen[i], klasse)
179                 liste.append(subregionenMit)
180                 start = start + len(subregionenMit)
181                 self.pos.append(start)
```

```
                #if time.time() - letzteAusgabe >= 1.0:
                #    sys.stderr.write('start'+" "+kname+"-Regionen zu " + rname + "\r")
                #    letzteAusgabe = time.time()
            #}
            #sys.stderr.write('start'+" "+kname+"-Regionen zu " + rname + ".\n")
            Regionen.__init__(self, liste)
        #}
    #}

    def __len__(self):
        retval = self.pos[len(self.pos)-1]
        if retval < 0:
            sys.stderr.write("RegionenMit: len < 0\n")
        return retval

    def getItemAtIndex(self, index):
        key = index
        pos = bisect.bisect(self.pos, key)-1
        try:
            return self.subregionen[pos][key - self.pos[pos]]
        except:
            raise IndexError

#}


class RegionenSlice(Regionen): #{

    def __repr__(self):
        return "<" + 'self.reg' + "[" + 'self.start' + \
                                  ":" + 'self.stop' + \
                                  ", " + 'self.step' + "]>"

    def __init__(self, reg, slice):
        self.reg = reg
        start, stop, step = (slice.start, slice.stop, slice.step)
        if not start:
            start = 0
        if not stop:
            stop = len(reg)
        if start < 0:
            start = start + len(reg)
        if start < 0:
            start = 0
        if stop < 0:
            stop = stop + len(reg)
        if stop < 0:
            stop = 0
        if stop > len(reg):
            stop = len(reg)
        if start > len(reg):
            start = len(reg)
        if start > stop:
            start = stop
        self.start = start
        self.stop  = stop
        self.step  = step
        if self.start != slice.start or self.stop != self.stop:
            sys.stderr.write("Warnung: Slice %s -> %s, da len=%s\n" %('slice','self','len(reg)'))
        if step:
            sys.stderr.write('self' + " noch nicht implementiert\n")
```

```
243
244     def __getstate__(self):
245         return (self.reg, self.start, self.stop, self.step)
246
247     def __setstate__(self, state):
248         self.reg, self.start, self.stop, self.step = state
249
250     def __len__(self):
251         retval = self.stop - self.start
252         if retval < 0:
253             sys.stderr.write("RegionenSlice: len < 0\n")
254         return retval
255
256     def getItemAtIndex(self, index):
257         key = index + self.start
258         if key >= self.stop:
259             raise IndexError
260         return self.reg[key]
261
262     def getDelimiter(self):
263         return self.reg.getDelimiter()
264
265 #}
266
267 from DatabaseAPI import cache
```

M.1.8 satz.py

```
1  #!/usr/bin/python
2
3  # Modul Region.satz
4
5  import atom
6  import region
7
8  class Satz(region.Regionen): #{
9
10   def __repr__(self):
11     return "<Satz: " + ‘self.subregionen‘+ ">"
12
13   def getDelimiter(self):
14     return "<SATZ>\n"
15
16   delimiter = "<SATZ>\n"
17
18 #}
19
20 class SatzVonZeilenliste(Satz): #{
21   def __init__(self, zeilenliste):
22     atomliste = []
23     for zeile in zeilenliste:
24       a = atom.AtomVonZeile(zeile)
25       if a.isOK():
26         atomliste.append(a)
27     Satz.__init__(self, atomliste)
28 #}
29
30 if 0: #{
31   a = "Äpfel\tNN\tApfel\n"
32   b = "Birnen\tNN\tBirne\n"
33   c = SatzVonZeilenliste([a,b])
```

```
  print "Satz [a,b]:", c, `c`
  r = c.getRegionenMit(atom.Atom)
  print r, `r`, len(r)
  for i in range(0, len(r)):
    print ("["+`i`+"]: " + `r[i]`)
  print "Lemmata:", r.getLemmata()
#}
```

M.1.9 segment.py

```
#!/usr/bin/python

# Modul Region.segment

# KOKS Studienprojekt 2000 - 2001
# Korpusbasierte Kollokationensuche

import alignment
import atom
import region
import satz
import string

class Segment(region.Regionen): #{

    def __repr__(self):
        return "<Segment: " + `self.subregionen`+ ">"

    def getDelimiter(self):
        return "<segmentgrenze>\n"

    delimiter = "<segmentgrenze>\n"

    def quelleninfo(self):
        return "Segment %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)

#}


class SegmentPaar(Segment, alignment.AlignedRegion): #{

    def __init__(self, seg1, seg2, qs, zs):
        alignment.AlignedRegion.__init__(self, seg1, seg2, qs, zs)
        Segment.__init__(self, seg1)

    def alignedRegion(self):    # liefert das alignte Segment
        return SegmentPaar(self.reg2, self.reg1, self.lang2, self.lang1)

#}


class SegmentVonZeilenliste(Segment): #{

  def __repr__(self):
    return "<Segment: " + string.strip(self.getText()[0:80]) + ">"

  def __init__(self, zeilenliste):
    satzliste = []
    pos = 0
    start = 0
    inhaltGelesen = 0
```

```
    for zeile in zeilenliste: #{
      if zeile[0:6] == "<SATZ>": #{
        satzliste.append(
                        satz.SatzVonZeilenliste(zeilenliste[start:pos]))
        start = pos + 1
        inhaltGelesen = 0
      #}
      else: #{
        inhaltGelesen = 1
      #}
      pos = pos + 1
    #}
    if inhaltGelesen: #{         # Inhalt nach dem letzten Delimiter?
      satzliste.append(
                        satz.SatzVonZeilenliste(zeilenliste[start:pos]))
    #}
    Segment.__init__(self, satzliste)
#}

if 0: #{
  a = "Äpfel\tNN\tApfel\n"
  b = "Birnen\tNN\tBirne\n"
  c = SegmentVonZeilenliste([a,b])
  print "Segment [a,b]:", c, `c`
  r = c.getRegionenMit(atom.Atom)
  print r, `r`, len(r)
  for i in range(0, len(r)):
    print ("["+`i`+"]: " + `r[i]`)
  print "Lemmata:", r.getLemmata()
#}
```

M.2 Church and Gale Aligner

Den Aligner von Church und Gale haben wir an einigen Stellen verändert.

M.2.1 align.c

```
/*
The following code is the core of align.
It is a C language program which inputs two text files, with one
token (word) per line. The text files contain a number of delimiter tokens.
There are two types of delimiter tokens: ''hard'' and ''soft''.
The hard regions (e.g., paragraphs) may not be changed, and there must be
equal numbers of them in the two input files.
The soft regions (e.g., sentences) may be deleted (1-0), inserted (0-1),
substituted (1-1), contracted (2-1), expanded (1-2), or merged (2-2)
as necessary so that the output ends up with the same number of soft regions.
The program generates two output files. The two output files contain an
equal number of soft regions, each on a line. If the -v command line option is
included, each soft region is preceded by its probability score.
*/

#include <malloc.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#ifdef POSIX
# include <unistd.h>
#else
```

```
# define HAVE_STRING_H 1
# include "getopt.h"
#endif

#define MAXINT (~0 > 0 ? ~0 : (unsigned) ~0 >> 1)

/*
  usage:
  align_regions -D '.PARA' -d '.End of Sentence' file1 file2

  outputs two files: file1.al & file2.al

  hard regions are delimited by the -D arg
  soft regions are delimited by the -d arg
*/

static int endRegionsWithDelimiter = 0;

#define dist(x,y) distances[(x) * ((ny) + 1) + (y)]
#define pathx(x,y) path_x[(x) * ((ny) + 1) + (y)]
#define pathy(x,y) path_y[(x) * ((ny) + 1) + (y)]
#define MAX_FILENAME 256
#define BIG_DISTANCE 2500

/* Dynamic Programming Optimization */
struct alignment {
    int x1;
    int y1;
    int x2;
    int y2;
    int d;
};

char *hard_delimiter = NULL;    /* -D arg */
char *soft_delimiter = NULL;    /* -d arg */
int verbose = 0;                                /* -v arg */

/* utility functions */
char *readchars(char *filename, int *len_ptr),
     **readlines(char *filename, int *len_ptr),
     **substrings(char *string, char *end, char delimiter, int *len_ptr);
void err(char *msg);

/*
  seq_align by Mike Riley

  x and y are sequences of objects, represented as non-zero ints, to be aligned.

  dist_funct(x1, y1, x2, y2) is a distance function of 4 args:

    dist_funct(x1, y1, 0, 0) gives cost of substitution of x1 by y1.
    dist_funct(x1, 0, 0, 0) gives cost of deletion of x1.
    dist_funct(0, y1, 0, 0) gives cost of insertion of y1.
    dist_funct(x1, y1, x2, 0) gives cost of contraction of (x1,x2) to y1.
    dist_funct(x1, y1, 0, y2) gives cost of expansion of x1 to (y1,y2).
    dist_funct(x1, y1, x2, y2) gives cost to match (x1,x2) to (y1,y2).

  align is the alignment, with (align[i].x1, align[i].x2) aligned
  with (align[i].y1, align[i].y2). Zero in align[].x1 and align[].y1
  correspond to insertion and deletion, respectively. Non-zero in
  align[].x2 and align[].y2 correspond to contraction and expansion,
```

```
  respectively. align[].d gives the distance for that pairing.

  The function returns the length of the alignment.
*/

int
seq_align(
    int *x, int *y, int nx, int ny,
    int (*dist_funct)(),
    struct alignment **align)
{
    int *distances, *path_x, *path_y, n;
    int i, j, oi, oj, di, dj, d1, d2, d3, d4, d5, d6, dmin;
    struct alignment *ralign;
    distances = (int *) malloc((nx + 1) * (ny + 1) * sizeof(int));
    path_x = (int *) malloc((nx + 1) * (ny + 1) * sizeof(int));
    path_y = (int *) malloc((nx + 1) * (ny + 1) * sizeof(int));
    ralign = (struct alignment *) malloc((nx + ny) * sizeof(struct alignment));
    for(j = 0; j <= ny; j++) {
        for(i = 0; i <= nx; i++) {
            d1 = i>0 && j>0 ? /* substitution */
                dist(i-1, j-1) + (*dist_funct)(x[i-1], y[j-1], 0, 0)
                    : MAXINT;
            d2 = i>0 ? /* deletion */
                dist(i-1, j) + (*dist_funct)(x[i-1], 0, 0, 0)
                    : MAXINT;
            d3 = j>0 ? /* insertion */
                dist(i, j-1) + (*dist_funct)(0, y[j-1], 0, 0)
                    : MAXINT;
            d4 = i>1 && j>0 ? /* contraction */
                dist(i-2, j-1) + (*dist_funct)(x[i-2], y[j-1], x[i-1], 0)
                    : MAXINT;
            d5 = i>0 && j>1 ? /* expansion */
                dist(i-1, j-2) + (*dist_funct)(x[i-1], y[j-2], 0, y[j-1])
                    : MAXINT;
            d6 = i>1 && j>1 ? /* melding */
                dist(i-2, j-2) + (*dist_funct)(x[i-2], y[j-2], x[i-1], y[j-1])
                    : MAXINT;

            dmin = d1;
            if(d2<dmin) dmin=d2;
            if(d3<dmin) dmin=d3;
            if(d4<dmin) dmin=d4;
            if(d5<dmin) dmin=d5;
            if(d6<dmin) dmin=d6;

            if(dmin == MAXINT) {
                dist(i,j) = 0;
            }
            else if(dmin == d1) {
                dist(i,j) = d1;
                pathx(i,j) = i-1;
                pathy(i,j) = j-1;
            }
            else if(dmin == d2) {
                dist(i,j) = d2;
                pathx(i,j) = i-1;
                pathy(i,j) = j;
            }
            else if(dmin == d3) {
                dist(i,j) = d3;
```

```
146                 pathx(i,j) = i;
147                 pathy(i,j) = j-1;
148             }
149             else if(dmin == d4) {
150                 dist(i,j) = d4;
151                 pathx(i,j) = i-2;
152                 pathy(i,j) = j-1;
153             }
154             else if(dmin == d5){
155                 dist(i,j) = d5;
156                 pathx(i,j) = i-1;
157                 pathy(i,j) = j-2;
158             }
159             else /* dmin == d6 */ {
160                 dist(i,j) = d6;
161                 pathx(i,j) = i-2;
162                 pathy(i,j) = j-2;
163             }
164         }
165     }
166     n = 0;
167     for(i=nx, j=ny ; i>0 || j>0 ; i = oi, j = oj) {
168         oi = pathx(i, j);
169         oj = pathy(i, j);
170         di = i - oi;
171         dj = j - oj;
172 
173         if(di == 1 && dj == 1) { /* substitution */
174             ralign[n].x1 = x[i-1];
175             ralign[n].y1 = y[j-1];
176             ralign[n].x2 = 0;
177             ralign[n].y2 = 0;
178             ralign[n++].d = dist(i, j) - dist(i-1, j-1);
179         }
180 
181         else if(di == 1 && dj == 0) { /* deletion */
182             ralign[n].x1 = x[i-1];
183             ralign[n].y1 = 0;
184             ralign[n].x2 = 0;
185             ralign[n].y2 = 0;
186             ralign[n++].d = dist(i, j) - dist(i-1, j);
187         }
188 
189         else if(di == 0 && dj == 1) { /* insertion */
190             ralign[n].x1 = 0;
191             ralign[n].y1 = y[j-1];
192             ralign[n].x2 = 0;
193             ralign[n].y2 = 0;
194             ralign[n++].d = dist(i, j) - dist(i, j-1);
195         }
196 
197         else if(dj == 1) { /* contraction */
198             ralign[n].x1 = x[i-2];
199             ralign[n].y1 = y[j-1];
200             ralign[n].x2 = x[i-1];
201             ralign[n].y2 = 0;
202             ralign[n++].d = dist(i, j) - dist(i-2, j-1);
203         }
204 
205         else if(di == 1) { /* expansion */
206             ralign[n].x1 = x[i-1];
```

```
            ralign[n].y1 = y[j-2];
            ralign[n].x2 = 0;
            ralign[n].y2 = y[j-1];
            ralign[n++].d = dist(i, j) - dist(i-1, j-2);
        }

        else /* di == 2 && dj == 2 */ { /* melding */
            ralign[n].x1 = x[i-2];
            ralign[n].y1 = y[j-2];
            ralign[n].x2 = x[i-1];
            ralign[n].y2 = y[j-1];
            ralign[n++].d = dist(i, j) - dist(i-2, j-2);
        }
    }
    *align = (struct alignment *) malloc(n * sizeof(struct alignment));
    for(i=0; i<n; i++)
        memcpy((*align) + (n-i-1), ralign + i, sizeof(struct alignment));
        free(distances);
        free(path_x);
        free(path_y);
        free(ralign);
        return(n);
}

/* Local Distance Function */
/* Returns the area under a normal distribution
   from -inf to z standard deviations */
double
pnorm(
    double z)
{
    double t, pd;
    t = 1/(1 + 0.2316419 * z);
    pd = 1 - 0.3989423 *
        exp(-z * z/2) *
            ((((1.330274429 * t - 1.821255978) * t
                + 1.781477937) * t - 0.356563782) * t + 0.319381530) * t;
    /* see Abramowitz, M., and I. Stegun (1964), 26.2.17 p. 932 */
    return(pd);
}

/* Return -100 * log probability that an English sentence of length
   len1 is a translation of a foreign sentence of length len2. The
   probability is based on two parameters, the mean and variance of
   number of foreign characters per English character.
*/

int
match(
    int len1, int len2)
{
    double z, pd, mean;
    double c = 1;
    double s2 = 6.8 ;

    if(len1==0 && len2==0) return(0);
    mean = (len1 + len2/c)/2;
    z = (c * len1 - len2)/sqrt(s2 * mean);

    /* Need to deal with both sides of the normal distribution */
    if(z < 0) z = -z;
```

```
    pd = 2 * (1 - pnorm(z));

    if(pd > 0) return((int)(-100 * log(pd)));
    else return(BIG_DISTANCE);
}

int
two_side_distance(
    int x1, int y1, int x2, int y2)
{
    int penalty21 = 230;        /* -100 * log([prob of 2-1 match] / [prob of 1-1 match]) */
    int penalty22 = 440;        /* -100 * log([prob of 2-2 match] / [prob of 1-1 match]) */
    int penalty01 = 450; /* -100 * log([prob of 0-1 match] / [prob of 1-1 match]) */

    if(x2 == 0 && y2 == 0)

        if(x1 == 0) /* insertion */
            return(match(x1, y1) + penalty01);

        else if(y1 == 0) /* deletion */
            return(match(x1, y1) + penalty01);


        else return (match(x1, y1)); /* substitution */

    else if(x2 == 0) /* expansion */
        return (match(x1, y1 + y2) + penalty21);

    else if(y2 == 0) /* contraction */
        return(match(x1 + x2, y1) + penalty21);

    else /* merger */
        return(match(x1 + x2, y1 + y2) + penalty22);
}

/* Functions for Manipulating Regions */

struct region {
    char **lines;
    int length;           /* number of lines in region */
};

void
print_region(
    FILE *fd,
    struct region *region,
    int score)
{
    char **lines, **end;

    lines = region->lines;
    end = lines + region->length;  /* points one entry behind array */
    for( ; lines < end ; lines++)  /* print out all the valid lines */
        fprintf(fd, "%s\n", *lines);
}

/*
 * length_of_a_region(region)
 * returns amount of memory needed for lines of region region
 * serves as a distance-measure for seq_align()
 */
```

```
329
330 int
331 length_of_a_region(
332     struct region *region)
333 {
334     int result;
335     char **lines, **end;
336
337     lines = region->lines;
338     end = lines + region->length;
339     result = end - lines;        /* number of lines in region */
340
341     for( ; lines < end; lines++) /* add total chars in region */
342         result += strlen(*lines);
343     return(result);
344 }
345
346 int *
347 region_lengths(
348     struct region *regions,
349     int n)
350 {
351
352     int i;
353     int *result;
354
355     result = (int *)malloc(n * sizeof(int));
356     if(result == NULL) err("malloc failed");
357
358     for(i = 0; i < n; i++)
359         result[i] = length_of_a_region(regions+i);
360     return(result);
361 }
362
363 struct region *
364 find_sub_regions(
365     struct region *region,
366     char *delimiter,
367     int *len_ptr)
368 {
369     struct region *result;
370     char **l, **lines, **end;
371     int n = 0;
372
373     lines = region->lines;
374     end = lines + region->length;
375
376     for(l = lines; l < end; l++)
377         if(delimiter && strcmp(*l, delimiter) == 0) n++;
378
379     if (verbose) {
380         if (delimiter)
381             fprintf(stderr, "Number of delimiters %s found in region %p: %d\n",
382                     delimiter, region, n);
383         else
384             fprintf(stderr, "NULL delimiter\n");
385     }
386
387     result = (struct region *)calloc(n+2, sizeof(struct region)); /* +1 should also work */
388     if(result == NULL) err("malloc failed");
389     *len_ptr = n;
```

```
    if (!endRegionsWithDelimiter)
      (*len_ptr)++;
    n = 0;
    result[0].lines = lines;
    for(l = lines; l < end; l++)
        if(delimiter && strcmp(*l, delimiter) == 0) {
            result[n].length = l - result[n].lines;
            result[n+1].lines = l+1;
            n++;
        }
    result[n].length = l - result[n].lines;
    if (!endRegionsWithDelimiter)
      n++;
    if(n != *len_ptr) {
        fprintf(stderr, "find_sub_regions: n = %d, *len_ptr = %d\n", n, *len_ptr);
        exit(2);
    }
    if (verbose) {
      fprintf(stderr, "returning %d regions: %p upto %p\n",
              *len_ptr, result, result+(*len_ptr)-1);
    }
    return(result);
}

/* Top Level Main Function */
int
main(
    int argc,
    char **argv)
{
    char **lines1, **lines2;
    int number_of_lines1, number_of_lines2;
    struct region *hard_regions1, *hard_regions2, *soft_regions1, *soft_regions2;
    struct region *hard_end1, *hard_end2, tmp;
    int number_of_hard_regions1;
    int number_of_hard_regions2;
    int number_of_soft_regions1;
    int number_of_soft_regions2;
    int *len1, *len2;
    int c, n, i, ix, iy, prevx, prevy;
    struct alignment *align, *a;
    FILE *out1, *out2;
    char filename[MAX_FILENAME];
    extern char *optarg;
    extern int optind;

    /* parse arguments */
    while((c = getopt(argc, argv, "ved:D:")) != EOF)
        switch(c) {
        case 'v':
            verbose = 1;
            break;
        case 'e':
            endRegionsWithDelimiter = 1;     /* force original behaviour */
            break;
        case 'd':
            soft_delimiter = strdup(optarg);
            break;
        case 'D':
            hard_delimiter = strdup(optarg);
            break;
```

```
451         default:
452             fprintf(stderr, "usage: align_regions [d (soft delimiter)] [D (hard delimiter)]\n");
453             exit(2);
454         }
455 
456     if(argc != optind + 2) err("wrong number of arguments");
457 
458     /* open output files */
459     sprintf(filename, "%s.al", argv[optind]);
460     out1 = fopen(filename, "w");
461     if(out1 == NULL) {
462         fprintf(stderr, "can't open %s\n", filename);
463         exit(2);
464     }
465 
466     sprintf(filename, "%s.al", argv[optind+1]);
467     out2 = fopen(filename, "w");
468     if(out2 == NULL) {
469         fprintf(stderr, "can't open %s\n", filename);
470         exit(2);
471     }
472 
473     lines1 = readlines(argv[optind], &number_of_lines1);
474     lines2 = readlines(argv[optind+1], &number_of_lines2);
475 
476     if(verbose)
477         fprintf(stderr, "Processing input file 1.\n");
478 
479     tmp.lines = lines1;
480     tmp.length = number_of_lines1;
481     hard_regions1 = find_sub_regions(&tmp, hard_delimiter, &number_of_hard_regions1);
482 
483     if(verbose)
484         fprintf(stderr, "Processing input file 2.\n");
485 
486     tmp.lines = lines2;
487     tmp.length = number_of_lines2;
488     hard_regions2 = find_sub_regions(&tmp, hard_delimiter, &number_of_hard_regions2);
489 
490     if(verbose) {
491         fprintf(out1, "Hard regions: %d\n", number_of_hard_regions1);
492         fprintf(out2, "Hard regions: %d\n", number_of_hard_regions2);
493     }
494 
495     if(number_of_hard_regions1 != number_of_hard_regions2) {
496         fprintf(stderr, "Hard regions in file 1: %d\n",number_of_hard_regions1);
497         fprintf(stderr, "Hard regions in file 2: %d\n",number_of_hard_regions2);
498         err("align_regions: input files do not contain the same number of hard regions");
499     }
500 
501     if(verbose)
502         fprintf(stderr, "Number of hard regions OK.\n");
503 
504     hard_end1 = hard_regions1 + number_of_hard_regions1;
505     hard_end2 = hard_regions2 + number_of_hard_regions2;
506 
507     for( ; hard_regions1 < hard_end1 ; hard_regions1++, hard_regions2++) {
508 
509         soft_regions1 = find_sub_regions(hard_regions1+0, soft_delimiter,
510             &number_of_soft_regions1);
511         soft_regions2 = find_sub_regions(hard_regions2+0, soft_delimiter,
```

```
            &number_of_soft_regions2);

        len1 = region_lengths(soft_regions1, number_of_soft_regions1);
        len2 = region_lengths(soft_regions2, number_of_soft_regions2);

        n = seq_align(len1, len2, number_of_soft_regions1, number_of_soft_regions2,
            two_side_distance, &align);

        prevx = prevy = ix = iy = 0;
        for(i = 0; i < n; i++) {
            a = &align[i];
            if(a->x2 > 0) ix++; else if(a->x1 == 0) ix--;
            if(a->y2 > 0) iy++; else if(a->y1 == 0) iy--;
            if(a->x1 == 0 && a->y1 == 0 && a->x2 == 0 && a->y2 == 0) {ix++; iy++;}
            ix++;
            iy++;
            if(verbose) {
                fprintf(out1, ".Score %d\n", a->d);
                fprintf(out2, ".Score %d\n", a->d);
            }

            for( ; prevx < ix; prevx++)
                print_region(out1, soft_regions1+prevx, a->d);
            if (((i+1) < n ) || endRegionsWithDelimiter)
                fprintf(out1, "%s\n", soft_delimiter);

            for( ; prevy < iy; prevy++)
                print_region(out2, soft_regions2+prevy, a->d);
            if (((i+1) < n ) || endRegionsWithDelimiter)
                fprintf(out2, "%s\n", soft_delimiter);
        }
        if (((hard_regions1+1) < hard_end1) || endRegionsWithDelimiter) {
            fprintf(out1, "%s\n", hard_delimiter);
            fprintf(out2, "%s\n", hard_delimiter);
        }
        free(align);
        free(soft_regions1);
        free(soft_regions2);
        free(len1);
        free(len2);
    }

    return 0;
}

/* Utility Functions */

void
err(
    char *msg)
{
    fprintf(stderr, "**ERROR**: %s\n", msg);
    exit(2);
}

/* return the size of a file */
long filesize(
    FILE *fd)
{
    long curpos, length;
    curpos = ftell(fd);          /* save the current location in the file */
```

```
    fseek(fd, 0L, SEEK_END);     /* seek to the end of the file */
    length = ftell(fd);          /* get the current offset into the file */
    fseek(fd, curpos, SEEK_SET); /* restore saved position */
    return length;
}

/* return the contents of the file as a string
   and stuff the length of this string into len_ptr */
char *
readchars(
    char *filename,
    int *len_ptr)
{
    FILE *fd;
    char *result;

    fd = fopen(filename, "r");
    if(fd == NULL) err("open failed");

    *len_ptr = filesize(fd);

    result = malloc(*len_ptr);
    if(result == NULL) err("malloc failed\n");

    if(fread(result, sizeof(char), *len_ptr, fd) != *len_ptr)
        err("fread failed");

    if(fclose(fd) == -1)
        err("fclose failed");

    return(result);
}

/* split string into a number of substrings delimited by a delimiter character
   return an array of substrings
   stuff the length of this array into len_ptr
   attention: leading delimiter characters are skipped
*/
char **
substrings(
    char *string, char *end, char delimiter,
    int *len_ptr)
{
    char *s, **result;
    int i = 0;

    while(string < end && *string == delimiter) string++;

    for(s = string; s < end; s++)
        if(*s == delimiter) i++;
    *len_ptr = i;

    result = (char **)malloc(sizeof(char *) * (i+1));
    if(result == NULL) err("malloc failed");

    i = 0;
    result[i++] = string;
    for(s = string; s < end; s++)
        if(*s == delimiter) {
            result[i++] = s+1;
            *s = 0;
```

```
        }
    i--;        /*the last entry is beyond the end*/
    if(i != *len_ptr) {
        fprintf(stderr, "align_regions: confusion; i= %d; *len_ptr = %d\n", i, *len_ptr);
        exit(2);
    }

    return(result);
}

/* return an array of strings, one string for each line of the file
   set len_ptr to the number of lines in the file */
char **
readlines(
    char *filename,
    int *len_ptr)
{
    char *chars;
    int number_of_chars;
    chars = readchars(filename, &number_of_chars);
    return(substrings(chars, chars + number_of_chars, '\n', len_ptr));
}
```

M.2.2 getopt.c und getopt.h

Für nicht POSIX Systeme ist align.c bereits darauf vorbereitet, dass ggf. die getopt Quellen eingebunden werden müssen. Den Quellcode des Moduls getopt kann der GNU C-Library glibc entnommen werden. Auf diese Weise wurde align.c mit dem Watcom C/C++ Compiler 10.6 unter MS Windows 95 mit getopt.[ch] aus der glibc 2.2.0 übersetzt. Das Makefile (siehe M.4.1) geht von einem POSIX System aus.

M.3 Weitere Python Module und Skripte

...

M.3.1 abstand.py

Siehe auch Beschreibung der Abstandsmaße ab Seite 55 und insbesondere Seite 62 wegen eines Fehlers im TG2 Abstandsmaß.

```
#!/usr/bin/python

# Modul abstand

import zeitmessung

import math
import popen2
import string
import StringIO
import sys

class Abstand: #{

    def __init__(self, default = 0.0):      # aufrufen wg. aiout
        self.default = default
        self.aiout=NullFile()

    def setAiout(self, out):
        self.aiout = out

```

```
    def __repr__(self):                     # diese Methoden ueberschreiben
        return "<Abstand immer %s>" %self.default

    def zwischen(self, reg1, reg2):         # diese Methode ueberschreiben
        self.aiout.write("Dummyabstand %s\n" %self.default)
        self.aiout.write("<SATZ>\n")
        return self.default

    def __call__(self, reg1, reg2):         # bequemer Aufruf
        return self.zwischen(reg1, reg2)

    def statistic(self):
        return "keine Statistik vorhanden"

    def close(self):
        pass

#}

class AbstandEQU(Abstand): #{

  def __init__(self):
    Abstand.__init__(self)

  def __repr__(self):
    return "<AbstandEQU>"

  def zwischen(self, reg1, reg2):
     m1 = map(None, reg1.getWoerter(),
                    reg1.getTags(),
                    reg1.getLemmata())
     m2 = map(None, reg2.getWoerter(),
                    reg2.getTags(),
                    reg2.getLemmata())
     if m1 == m2:
       self.aiout.write("identisch\n<SATZ>\n")
       return 0.0
     self.aiout.write("verschieden\n<SATZ>\n")
     return 1.0

#}


class AbstandChurchAndGale(Abstand): #{

  def __repr__(self):
    return "<AbstandChurchAndGale>"

  def __init__(self, c = 1.0, s2 = 6.8):
    Abstand.__init__(self)
    self.c = c
    self.s2 = s2

  def zwischen(self, reg1, reg2):
    len1 = len(reg1.getText())
    len2 = len(reg2.getText())
    return self.p_zwischen(len1, len2)

  def p_zwischen(self, len1, len2):
    if len1==0 and len2==0:
      self.aiout.write("Laengen null\n<SATZ>\n")
```

```
      return 0
    c = self.c
    s2 = self.s2
    mean = (len1 + len2/c)/2
    z = (c * len1 - len2)/math.sqrt(s2 * mean)    # sqrt(s2 * len1) ?
    self.aiout.write("z = %s\n" %‘z‘)
    if z < 0:
      z = -z
    pd =  2 * (1 - self.pnorm(z))             # uebernommen aus align.c
    self.aiout.write("<SATZ>\n")
    return 1.0 - pd
    #if pd<0.00000001:
    #  pd= 0.00000001
    #return -math.log(pd)

  def pnorm(self, z):                         # uebernommen aus align.c
    "Returns the area under a normal distribution"
    "from -inf to z standard deviations"
    t = 1.0/(1.0 + 0.2316419 * z)
    pd = 1.0 - 0.3989423 * \
        math.exp(-z * z/2) * \
            ((((1.330274429 * t - 1.821255978) * t \
                + 1.781477937) * t - 0.356563782) * t + 0.319381530) * t
    # see Abramowitz, M., and I. Stegun (1964), 26.2.17 p. 932
    return pd
#}

class AbstandTest(Abstand): #{

  def __init__(self):
    Abstand.__init__(self)

  def __repr__(self):
    return "<AbstandTest>"

  def zwischen(self, reg1, reg2):
    len1 = len(reg1.getText())
    len2 = len(reg2.getText())
    d = len1 - len2
    self.aiout.write("d = %s\n" %‘d‘)
    if d < 0:
      d = -d
    if d > 10:
      d = 10
    self.aiout.write("<SATZ>\n")
    return 0.1 * d

#}

class AbstandBidirectional(Abstand):

    def __repr__(self):
        return "<AbstandBidirectional>"

    def __init__(self, d2e, e2d):
        Abstand.__init__(self)
        self.d2e = d2e
        self.e2d = e2d

    def close(self):
        self.d2e.close()
```

```
        self.e2d.close()

    def zwischen(self, reg1, reg2):
        a = self.d2e.zwischen(reg1, reg2)
        b = self.e2d.zwischen(reg2, reg1)
        self.aiout.write("Richtung d2e = %s\n" %‘a‘)
        self.aiout.write("Richtung e2d = %s\n" %‘b‘)
        self.aiout.write("<SATZ>\n")
        return 0.5 * a + 0.5 * b


class AbstandWB(AbstandBidirectional):

    def __repr__(self):
        return "<AbstandWB>"

    def __init__(self):
        d2e = AbstandWBUnidirectional(’de’, ’en’)
        e2d = AbstandWBUnidirectional(’en’, ’de’)
        AbstandBidirectional.__init__(self, d2e, e2d)


class AbstandWBUnidirectional(Abstand): #{

  def __repr__(self):
    return "<AbstandWBUnidirectional>"

  def __init__(self, quellsprache, zielsprache):
    Abstand.__init__(self)
    self.quellsprache = quellsprache
    self.zielsprache  = zielsprache
    relevant1 = filterwb.RelevantNachTagP(zielsprache,0,1)  # max = 1
    self.wbf = filterwb.WoerterbuchFilter(quellsprache, zielsprache, relevant1)
    self.relevant = filterwb.RelevantNachTagP(zielsprache)
    self.atg = AbstandTG_2(3)

  def close(self):
    self.wbf.close()
    self.atg.close()

  def p_reduzierteWoerter(self, wliste):
    liste = []
    for wort in wliste:
      nwort = string.lower(wort)
      nwort = string.replace(nwort, ".", "")
      nwort = string.replace(nwort, ",", "")
      nwort = string.replace(nwort, "-", "")
      liste.append(nwort)
    return liste

  def zwischen(self, reg1, reg2):
    wb = self.wbf[reg1]
    a1 = reg1.getRegionenMit(atom.Atom)
    a2 = reg2.getRegionenMit(atom.Atom)
    l1 = reg1.getLemmata()
    l2 = reg2.getLemmata()
    w1 = reg1.getWoerter()
    w2 = reg2.getWoerter()
    wr1 = self.p_reduzierteWoerter(w1)
    wr2 = self.p_reduzierteWoerter(w2)
    uebrig1 = len(a1) * [1]
```

```
    uebrig2 = len(a2) * [1]
    c1 = {}
    c2 = {}
    if l1 == []:
      self.aiout.write("Region 1 leer.\n<SATZ>\n")
      return 0.0
    for index in range(0, len(a1)): #{
      if wb.has_key(index): #{               # Relevant?
        lemma = l1[index]
        if lemma != "<unknown>": #{
          c1[lemma] = l1.count(lemma)
          ueabsatz = wb[index]
          count2 = 0
          if len(ueabsatz) > 0:
            uebrig1[index] = 0
            uemenge = {}
            ueatome = ueabsatz.getRegionenMit(atom.Atom)
            for ueatom in ueatome:
              if self.relevant.relevant(ueatom):
                ueatomLemma = ueatom.getLemmata()[0]
                if ueatomLemma != "<unknown>":
                  uemenge[ueatomLemma] = 1
            self.aiout.write('"'+lemma+'" -> "'+string.join(uemenge.keys(),'", "')+'"\n')
            for ue in uemenge.keys():
              try:
                pos = l2.index(ue)
                uebrig2[pos] = 0
                count2 = count2 + 1
              except ValueError:
                pass
          #}
          else: #{          # Sonderbehandlung: Wort anstatt Lemma messen
            try:
              pos = wr2.index(wr1[index])
              uebrig2[pos] = 0
              count2 = count2 + 1
            except ValueError:
              pass
            #minimum = 1.0
            #for i in range(0, len(a2)):
            #   temp = self.atg(a1[index], a2[i])
            #   if temp < minimum:
            #     minimum = temp
            #count2 = 1.0 - temp
          #}
          c2[lemma] = count2
        #}
      #}
    #}
    uebrig1atome = []
    for index in range(0, len(a1)):
      if uebrig1[index]:
        uebrig1atome.append(a1[index])
    uebrig1atome = region.Regionen(uebrig1atome)
    uebrig2atome = []
    for index in range(0, len(a2)):
      if uebrig2[index]:
        uebrig2atome.append(a2[index])
    uebrig2atome = region.Regionen(uebrig2atome)
    atgAbstand = self.atg(uebrig1atome, uebrig2atome)
    atgAnteil = len(uebrig1atome)
```

```
    insgesamt = len(a1)
    awbAnteil = insgesamt - atgAnteil
    x = 0.0
    h2 = 0
    for i in c1.keys():
      t = float(c2[i]) / float(c1[i])
      x = x + t
      self.aiout.write(‘(i, c1[i], c2[i], t)‘+’\n’)
      if c2[i]>0:
        h2 = h2 + 1
    n = len(c1.keys())
    if n == 0:
        self.aiout.write("Region 1 hat keine relevanten Woerter.\n<SATZ>\n")
        return 0.0
    x = x / n
    self.aiout.write("Mittelwert " + ‘x‘ +  " - benutze aber 1.0"+’\n’)
    x = 1.0
    v = 0.0
    for i in c1.keys():
      t = (float(c2[i]) / float(c1[i]))
      t = t - x
      v = v + t * t
    v = v / n
    #2do Kennzahl v umrechnen in Wahrscheinlichkeit fuer Entsprechung
    # von reg1 und reg2
    awbAbstand = 1.0 - math.exp(-v)
    awbAbstand = 1.0 - float(h2) / float(len(c1.keys()))
    self.aiout.write("atg " + ‘atgAnteil‘ + ‘atgAbstand‘+’\n’)
    self.aiout.write("awb " + ‘awbAnteil‘ + ‘awbAbstand‘+’\n’)
    self.aiout.write("<SATZ>\n")
    return (atgAnteil * atgAbstand + awbAnteil * awbAbstand) / float(insgesamt)


class AbstandTG(Abstand): #{

  def __init__(self, laenge = 3):
    Abstand.__init__(self)
    self.laenge = laenge

  def __repr__(self):
    return "<AbstandTG>"

  def zwischen(self, reg1, reg2):
    t1 = " " + reg1.getText() + " "
    t2 = " " + reg2.getText() + " "
    # self.aiout.write(("---"+‘reg1.getText()‘+"/"+‘reg2.getText()‘+"---")+’\n’)
#    self.aiout.write(("---"+‘t1‘+"/"+‘t2‘+"---")+’\n’)
    if t1 == "":
      self.aiout.write("leere Quellregion"+’\n’)
      self.aiout.write("<SATZ>\n")
      return 0.0
    # self.aiout.write(("---"+‘t1‘+"/"+‘t2‘+"---")+’\n’)
    c1 = self.counttg(t1)
    c2 = self.counttg(t2)
    self.aiout.write("c1:" + ‘c1‘ + ’\n’)
    self.aiout.write("c2:" + ‘c2‘ + ’\n’)
    relevant = c1.keys()
    for i in c1.keys():
      if not c2.has_key(i):
        c2[i] = 0
    x = 0.0
```

```
    h2 = 0
    for i in relevant:
      t = float(c2[i]) / float(c1[i])
      x = x + t
      self.aiout.write('i' + '\t' + 'c1[i]' + '\t' + 'c2[i]' + '\t' + '\n')
      if c2[i] > 0:
        h2=h2+1
    n = len(relevant)
    if n == 0:
        self.aiout.write("keine %s-gramme in Region 1\n<SATZ>\n" %self.laenge)
        return 0.0
    #x = x / n
    x = 1.0
    v = 0.0
    for i in relevant:
      t = (float(c2[i]) / float(c1[i]))
      t = t - x
      v = v + t * t
    v = v / n
    #2do Kennzahl v umrechnen in Wahrscheinlichkeit fuer Entsprechung
    # von reg1 und reg2
    atgAbstand = 1.0 - math.exp(-v)
    # atgAbstand = 1.0 - float(h2) / float(len(relevant))
    self.aiout.write("<SATZ>\n")
    return atgAbstand

  def counttg(self, text):
    c = {}
    for i in range(0, len(text)-self.laenge+1):
      tg = text[i:i+self.laenge]
      if not c.has_key(tg):
        c[tg] = 1
      else:
        c[tg] = c[tg] + 1
    return c

#}

class AbstandTG_2(Abstand): #{

    def __init__(self, laenge = 3):
        Abstand.__init__(self)
        self.laenge = laenge
        self.cache = cache.DefaultCache(self)

    def __repr__(self):
        return "<AbstandTG_2>"

    def p_reduziert(self, wort):
        nwort = string.lower(wort)
        for i in [".", ",", ":", "$"]:
            nwort = string.replace(nwort, i, "")
        for i in ["-", "/"]:
            nwort = string.replace(nwort, i, " ")
        return " " + nwort + " "

    def zwischen(self, reg1, reg2):
        t1 = self.p_reduziert(reg1.getText())
        t2 = self.p_reduziert(reg2.getText())
        c1 = self.counttg(t1)
        c2 = self.counttg(t2)
```

```
388         key1 = c1.keys()
389         key2 = c2.keys()
390         last = ""
391         keys = key1 + key2
392         keys.sort()
393         gemeinsam = 0
394         insgesamt = 0
395         for key in keys:
396             if last == key:
397                 continue
398             if c1.has_key(key):
399                 num1 = c1[key]
400             else:
401                 num1 = 0
402             if c2.has_key(key):
403                 num2 = c2[key]
404             else:
405                 num2 = 0
406             gemeinsam = gemeinsam + min([num1,num2])
407             insgesamt = insgesamt + max([num1,num2])
408             last = key   # siehe Doku im Abschlussbericht
409         if insgesamt > 0:
410             v = float(gemeinsam) / float(insgesamt)
411         else:
412             v = 1.0
413         atgAbstand = 1.0 - v
414         self.aiout.write("gemeinsam,insgesamt = %s\n" %‘(gemeinsam, insgesamt)‘)
415         self.aiout.write("’%s’ - ’%s’\n" %(t1,t2))
416         self.aiout.write("<SATZ>\n")
417         return atgAbstand
418 
419     def counttg(self, text):
420         ckey = text
421         if self.cache.has_key(ckey):
422             retval = self.cache[ckey]
423             # sys.stderr.write("TG_2: %s aus cache\n" %‘ckey‘)
424         else:
425             c = {}
426             for i in range(0, len(text)-self.laenge+1):
427                 tg = text[i:i+self.laenge]
428                 if not c.has_key(tg):
429                     c[tg] = 1
430                 else:
431                     c[tg] = c[tg] + 1
432             retval = c
433             self.cache[ckey] = retval
434         return retval
435 
436 
437 class NullFile:
438     def write(self, data):
439         pass
440 
441 class AbstandWB_2(Abstand, zeitmessung.Zeitmessung):
442 
443     def __repr__(self):
444         if self.nurNomen:
445             s = "<AbstandWB_2 %s nach %s (nur Nomen)>"
446         else:
447             s = "<AbstandWB_2 %s nach %s>"
448         return s %(self.quellsprache,self.zielsprache)
```

```
449
450     def __init__(self, quellsprache = "de", zielsprache = "en", nurNomen = 0):
451         zeitmessung.Zeitmessung.__init__(self)
452         Abstand.__init__(self)
453         self.quellsprache = quellsprache
454         self.zielsprache  = zielsprache
455         if nurNomen:
456             temp = filterwb.RelevantWegenNomen(quellsprache,0,1)  # max = 1
457         else:
458             temp = filterwb.RelevantNachTagP(quellsprache, 0, 1)
459         self.wbf1 = filterwb.WoerterbuchFilter(quellsprache, zielsprache, temp)
460         if nurNomen:
461             temp = filterwb.RelevantWegenNomen(zielsprache,0,1)   # max = 1
462         else:
463             temp = filterwb.RelevantNachTagP(zielsprache, 0, 1)
464         self.wbf2 = filterwb.WoerterbuchFilter(zielsprache, quellsprache, temp)
465         self.nurNomen = nurNomen
466         self.atg = AbstandTG_2(3)
467         self.acg = AbstandChurchAndGale()
468         self.cache = cache.DefaultCache(self)
469
470     def close(self):
471         self.wbf1.close()
472         self.wbf2.close()
473         self.atg.close()
474         self.acg.close()
475
476     def statistic(self):
477         return { "wbf1" : self.wbf1.statistic(),
478                  "wbf2" : self.wbf2.statistic(),
479                  "wb1"  : self.wbf1.wb.statistic(),
480                  "wb2"  : self.wbf2.wb.statistic(),
481                  "self" : zeitmessung.Zeitmessung.statistic(self),
482                }
483
484     def p_daten(self, reg1, wbf1):
485         self.p_begin("p_daten")
486         ckey = (‘wbf1‘,reg1.getText())                    # 2do
487         if self.cache.has_key(ckey):
488             retval = self.cache[ckey]
489             # sys.stderr.write("WB_2: %s aus cache\n" %‘ckey‘)
490         else:
491             wb1 = wbf1[reg1]
492             a1 = reg1.getRegionenMit(atom.Atom)
493             l1 = reg1.getLemmata()
494             w1 = reg1.getWoerter()
495             wr1 = None  # self.p_reduzierteWoerter(w1)
496             i1 = len(a1) * [0]                        # 0 = ’zu bearbeiten’
497             relevant1 = []
498             for index in range(len(i1)):
499                  if wb1.has_key(index):               # relevant?
500                      #lemma1 = l1[index]
501                      relevant1.append(index)
502                  else:
503                      i1[index] = 1                    # als irrelevant markieren
504             retval = (wb1, a1, l1, w1, wr1, i1, relevant1)
505             self.cache[ckey] = retval
506         (wb1, a1, l1, w1, wr1, i1, relevant1) = retval
507         i1 = i1[:]                                      # Kopie erstellen
508         self.p_end("p_daten")
509         return (wb1, a1, l1, w1, wr1, i1, relevant1)
```

```
510
511     def zwischen(self, reg1, reg2):
512         self.p_begin("zwischen")
513         (wb1, a1, l1, w1, wr1, i1, relevant1) = self.p_daten(reg1, self.wbf1)
514         (wb2, a2, l2, w2, wr2, i2, relevant2) = self.p_daten(reg2, self.wbf2)
515         aiextra1 = StringIO.StringIO()
516         aiextra2 = StringIO.StringIO()
517         temp = self.aiout
518         self.aiout = aiextra1
519         self.p_setUe(relevant1, relevant2, wb1, l1, l2, i1, i2)
520         self.aiout = aiextra2
521         self.p_setUe(relevant2, relevant1, wb2, l2, l1, i2, i1)
522         self.aiout = temp
523         self.aiout.write("info1 = %s\n" %‘map(None, i1, w1)‘)
524         self.aiout.write("info2 = %s\n" %‘map(None, i2, w2)‘)
525         uenum1 = self.p_uenum(relevant1, i1)
526         uenum2 = self.p_uenum(relevant2, i2)
527         gemeinsam = min([uenum1, uenum2])
528         insgesamt = max([uenum1, uenum2])
529         self.aiout.write("AWB_2: g,i = %s\n" %‘(gemeinsam,insgesamt)‘)
530         if insgesamt > 0:
531             awbAbstand = 1.0 - float(gemeinsam) / float(insgesamt)
532         else:
533             awbAbstand = 1.0   # 2do: Bewertung von ’insgesamt’ verbessern
534         awbAnteil  = uenum1 + uenum2
535         uebrig1 = self.p_uebrig(a1, relevant1, i1)
536         uebrig2 = self.p_uebrig(a2, relevant2, i2)
537         self.aiout.write("uebrig1 = %s\n" %uebrig1.getText())
538         self.aiout.write("uebrig2 = %s\n" %uebrig2.getText())
539         atgAbstand = self.atg(uebrig1, uebrig2)
540         atgAnteil = len(uebrig1) + len(uebrig2)
541         regcg1 = len(i1) - uenum1 - len(uebrig1)
542         if regcg1 < 0:
543             sys.stderr.write("Fehler: |i1|-uenum1-|uebrig1| = %s\n" %regcg1)
544         regcg2 = len(i2) - uenum2 - len(uebrig2)
545         if regcg2 < 0:
546             sys.stderr.write("Fehler: |i2|-uenum2-|uebrig2| = %s\n" %regcg1)
547         acgAnteil = regcg1 + regcg2
548         acgAbstand = self.acg.p_zwischen(regcg1, regcg2)    # Hack
549         # 2do: eigentlich muesste man nicht die Anzahl der Woerter
550         # sondern die Anzahl der Zeichen an acg uebergeben
551         self.aiout.write("len(regcg1) = %s\n" %regcg1)
552         self.aiout.write("len(regcg2) = %s\n" %regcg2)
553         self.aiout.write("WB: wert, anteil = %s\n" %‘(awbAbstand,awbAnteil)‘)
554         self.aiout.write("TG: wert, anteil = %s\n" %‘(atgAbstand,atgAnteil)‘)
555         self.aiout.write("CG: wert, anteil = %s\n" %‘(acgAbstand,acgAnteil)‘)
556         bonus = max([len(i1),len(i2)])
557         if awbAnteil > 0:
558             awbAnteil = awbAnteil + bonus
559             if atgAnteil > 0:
560                 awbAnteil = awbAnteil + bonus
561             self.aiout.write("WB-Anteil erhoeht auf %s\n" %awbAnteil)
562         if atgAnteil > 0:
563             atgAnteil = atgAnteil + bonus
564             self.aiout.write("TG-Anteil erhoeht auf %s\n" %atgAnteil)
565         gesamt = awbAnteil + atgAnteil + acgAnteil
566         if gesamt > 0:
567             retval = float( awbAnteil * awbAbstand +\
568                             atgAnteil * atgAbstand +\
569                             acgAnteil * acgAbstand    ) / float( gesamt )
570         else:
```

```
            retval = 0.0
        self.aiout.write("Abstand insgesamt = %s\n" %retval)
        aiextra1.seek(0)
        self.aiout.write("--- Ue-Info (1):\n")
        self.aiout.write(aiextra1.read())
        aiextra2.seek(0)
        self.aiout.write("--- Ue-Info (2):\n")
        self.aiout.write(aiextra2.read())
        self.aiout.write("\n<SATZ>\n")
        self.p_end("zwischen")
        return retval

    def p_uebrig(self, atome, relevant, info):
        self.p_begin("p_uebrig")
        uebrig = []
        for index in relevant:
            if (info[index] & 2) == 0:      # noch nicht bearbeitet?
                uebrig.append(atome[index])
        retval = region.Regionen(uebrig)
        self.p_end("p_uebrig")
        return retval

    def p_setUe(self, relevant1, relevant2, wb1, l1, l2, i1, i2):
        self.p_begin("p_setUe")
        for index1 in relevant1:
            if wb1.has_key(index1):              # relevant?
                lemma1 = l1[index1]
                ue1 = wb1[index1]
                ue1lemmata = ue1.getLemmata()    # leer wenn lemma1 '<unknown>'
                #if not ue1lemmata:               # leer?   # 2do
                #    continue                     # dann aussichtslos
                for index2 in relevant2:
                    lemma2 = l2[index2]
                    if lemma2 == "<unknown>":
                        pass
                        # 2do: Woerter vergleichen
                        # wort2[index2] in ue1.getWoerter() ?
                    elif lemma2 in ue1lemmata:
                        i1[index1] = i1[index1] | 2   # markieren
                        i2[index2] = i2[index2] | 2
                        self.aiout.write("%s -> %s\n" %(lemma1,ue1lemmata))
                        self.aiout.write("index %s -> %s\n" %(index1,index2))
            #else:  # 2do: klaeren, ob relevant1 nich in p_daten entspr. init.
            #    sys.stderr.write("if doch notwendig\n")
        self.p_end("p_setUe")

    def p_uenum(self, relevant, info):    # zaehlt die markierten Woerter
        self.p_begin("p_uenum")
        num = 0
        for index in relevant:
             if (info[index] & 2) != 0:
                 num = num + 1
        self.p_end("p_uenum")
        return num

import wb
import filterwb
import eingabe
from DatabaseAPI import  config, db, regiondb, cache
from Region      import  atom, absatz, region
```

```
def main():
    dbconn = db.Db()
    regiondb.setDefaultDbconn(dbconn)
    ab = { "C&G"   : AbstandChurchAndGale(),
           "CG"    : AbstandChurchAndGale(),
           "TEST"  : AbstandTest(),
           "NULL"  : Abstand(0.0),
           "EINS"  : Abstand(1.0),
           "WB"    : AbstandWB(),
           "TG"    : AbstandTG(),
           "WB2"   : AbstandWB_2(),
           "TG2"   : AbstandTG_2(),
           "WB2N"  : AbstandWB_2("de","en",1),  # nur Nomen
         }
    wahl = "WB2"
    fout = sys.stdout
    verbose = 0
    text = 0
    hilfe = len(sys.argv) == 1
    reg = "segment"
    dok = eingabe.Liste(verbose)
    while len(sys.argv) > 1: #{
        if verbose:
            sys.stderr.write("args left: %s\n" %‘sys.argv[1:]‘)
        if sys.argv[1] == "-h":
            hilfe = 1
            break
        elif sys.argv[1] == "-u":
            text = 1
            del sys.argv[1]
        elif sys.argv[1] == "-v":
            verbose = 1
            dok.setVerbose(verbose)
            del sys.argv[1]
        elif sys.argv[1][:2] == "-r":
            reg = sys.argv[1][2:]
            del sys.argv[1]
            if not reg:
                try:
                    reg = sys.argv[1]
                    del sys.argv[1]
                except:
                    hilfe = 1
                    break
            if verbose:
                sys.stderr.write("region type = %s\n" %wahl)
        elif sys.argv[1][:2] == "-m":
            wahl = sys.argv[1][2:]
            del sys.argv[1]
            if not wahl:
                try:
                    wahl = sys.argv[1]
                    del sys.argv[1]
                except:
                    hilfe = 1
                    break
            if verbose:
                sys.stderr.write("distance measure = %s\n" %wahl)
        elif sys.argv[1][:2] == "-o":
            fname = sys.argv[1][2:]
            del sys.argv[1]
```

```
693             if not fname:
694                 try:
695                     fname = sys.argv[1]
696                     del sys.argv[1]
697                 except:
698                     hilfe = 1
699                     break
700             fout = open(fname, "w")
701             if verbose:
702                 sys.stderr.write("output to file %s\n" %fname)
703         else:
704             eaten = dok.eatOptions(sys.argv)
705             if not eaten:
706                 hilfe = 1
707                 break
708             if verbose:
709                 sys.stderr.write("eingabe.Liste has eaten %s args\n" %eaten)
710     if hilfe:
711         if verbose:
712             sys.stderr.write("args left: %s\n" %‘sys.argv[1:]‘)
713         sys.stderr.write("Usage: %s <options>\n" %sys.argv[0])
714         sys.stderr.write("options:  -h           print out this help\n")
715         sys.stderr.write("          -m distmeas  choose distance measure:\n")
716         keys = ab.keys()
717         keys.sort()
718         for w in keys:
719             sys.stderr.write("                          %s\t%s\n" %(w,‘ab[w]‘))
720         sys.stderr.write("          -o filename  write output to file\n")
721         sys.stderr.write("          -r regtype   one of\n")
722         for i in ["satz", "segment", "absatz", "dokument"]:
723             sys.stderr.write(25*’ ’ + "%s\n" %i)
724         sys.stderr.write("          -u           output index and text\n")
725         sys.stderr.write("          -v           verbose mode\n")
726         dok.help(sys.stderr)
727         dok.close()
728         dbconn.close()
729         for key in ab.keys():
730             ab[key].close()
731         sys.exit(1)
732     if verbose:
733         sys.stderr.write("distance measure %s = %s\n" %(wahl,‘ab[wahl]‘))
734     dok.finishOptions()
735     if   dok.getQuellsprache() == "de" and dok.getZielsprache() == "en":
736         l1 = eingabe.getRegionen(dok, reg)
737         l2 = eingabe.getRegionen(dok.alignedRegion(), reg)
738     elif dok.getQuellsprache() == "en" and dok.getZielsprache() == "de":
739         l1 = eingabe.getRegionen(dok.alignedRegion(), reg)
740         l2 = eingabe.getRegionen(dok, reg)
741     elif not wahl in ["WB"]:       # Liste der sprachabhaengigen Abstandsmasse
742         l1 = eingabe.getRegionen(dok, reg)                # reihenfolge egal
743         l2 = eingabe.getRegionen(dok.alignedRegion(), reg)
744     else:
745         sys.stderr.write("Error: cannot compare %s " %dok.getQuellsprache())
746         sys.stderr.write("and %s " %dok.getZielsprache())
747         sys.stderr.write("with %s\n" %‘ab[wahl]‘)
748         sys.exit(1)
749     if verbose:
750         sys.stderr.write("number of regions: %s\n" %‘(len(l1),len(l2))‘)
751     for index in range(max(len(l1), len(l2))):
752         if text:
753             fout.write("--- %s Nr. %s ---\n" %(reg, index))
```

```
            if l1.has_key(index):
                fout.write(l1[index].getText()+'\n')
            if l2.has_key(index):
                fout.write(l2[index].getText()+'\n')
        if l1.has_key(index) and l2.has_key(index):
            wert = ab[wahl].zwischen(l1[index], l2[index])
            fout.write("%s\n" %`wert`)
        else:
            fout.write("***\n")
        if text:
            fout.write('\n')
    dok.close()
    for key in ab.keys():
        ab[key].close()
    dbconn.close()

if __name__ == '__main__':
    main()
```

M.3.2 align.sh

```
#!/bin/bash

# align.sh

# Aufruf: align.sh dokumentprefix abstandsmass
#   oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en

ALIGN=`dirname $0`

if [ "$7" ] ; then

    #MASS="TG2"
    MASS="WB2"
    #LEAVE_OUT_DIST=2.5
    #LEAVE_OUT_DIST_MAX=2.5
    #LEAVE_OUT_DIST_GRADIENT=0.00
    LEAVE_OUT_DIST=1.5
    LEAVE_OUT_DIST_MAX=3.53
    LEAVE_OUT_DIST_GRADIENT=0.12
    test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
    test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
    test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
    test "$3" = "<dokende>"        && HDEL="dokument"
    test "$3" = "<DOKENDE>"        && HDEL="dokument"
    test "$3" = "<ABSATZ>"         && HDEL="absatz"
    test "$5" = "<ABSATZ>"         && SDEL="absatz"
    test "$5" = "<segmentgrenze>"  && SDEL="satz"
    test "$5" = "<SATZ>"           && SDEL="satz"
    test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
    test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }

    RED1=`dirname "$6"`/`basename "$6" .de.tag`
    RED2=`dirname "$7"`/`basename "$7" .en.tag`
    STAMM="$6"
    test "$RED1" = "$RED2" && STAMM="$RED1"

    # Version mit Abstandsmatrix

    python "$ALIGN/nmatrix.py" -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_D

```

```
elif [ "$2" ] ; then

    python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/mergemtr

else

    echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
    echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
    exit 1

fi

```

M.3.3 align2al.py

```
#!/usr/bin/python

import os
import string
import sys
import types

from DatabaseAPI import dokumentdb, db, regiondb, config
from Region      import satz, dokument, segment, absatz

dbconn = db.Db()
regiondb.setDefaultDbconn(dbconn)

def main():
    global dbconn
    verbose = 0
    ausgabeInsArbeitsverzeichnis = 0
    hilfe = 0
    while len(sys.argv) > 1: #{
        if verbose:
            sys.stderr.write("args left: %s\n" %‘sys.argv[1:]‘)
        if sys.argv[1] == "-h":
            hilfe = 1
            break
        elif sys.argv[1] == "-v":
            verbose = 1
            del sys.argv[1]
        elif sys.argv[1] == "-t":
            ausgabeInsArbeitsverzeichnis = 1
            del sys.argv[1]
        else:
            break
    if not hilfe and len(sys.argv) == 2:
        fname = sys.argv[-1]
        if fname == "-":
            align = sys.stdin
        else:
            align = open(sys.argv[-1], "r")
    else:
        hilfe = 1
    if hilfe:
        sys.stderr.write("Benutzung: %s [-t] <.align-Datei>\n" %sys.argv[0])
        sys.stderr.write("\t-t = Ausgabe ins Arbeitsverzeichnis\n")
    else:
        zeile = align.readline()
        if   zeile[:14] == "Dokument-DB-ID":
```

```
            id = int(zeile[14:])
            dok1 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["de"])
            dok2 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["en"])
            out1 = open("db_"+`id`+".de.tag.al", "w")
            out2 = open("db_"+`id`+".en.tag.al", "w")
        elif zeile[:14] == "Dokument-Datei":
            name = string.strip(zeile[15:])
            dok1 = dokument.DokumentVonDatei(name+".de.tag")
            dok2 = dokument.DokumentVonDatei(name+".en.tag")
            if ausgabeInsArbeitsverzeichnis:
                name = os.path.basename(name)
            out1 = open(name+".de.tag.al", "w")
            out2 = open(name+".en.tag.al", "w")
        elif zeile[:19] == "Paar Dokument-Datei":
            pos = string.find(zeile, " und Dokument-Datei")
            name1 = string.strip(zeile[20:pos])
            name2 = string.strip(zeile[pos+20:])
            dok1 = dokument.DokumentVonDatei(name1)
            dok2 = dokument.DokumentVonDatei(name2)
            if ausgabeInsArbeitsverzeichnis:
                name1 = os.path.basename(name1)
                name2 = os.path.basename(name2)
            out1 = open(name1+".al", "w")
            out2 = open(name2+".al", "w")
        elif zeile[:6] == "Absatz":
            raise NotImplementedError
        else:
            raise Exception, "Dokumenttyp nicht identifizierbar: %s" %`zeile`
        zeile = align.readline()
        if string.lower(zeile) == "nach satz\n":
            l1 = dok1.getRegionenMit(satz.Satz)
            l2 = dok2.getRegionenMit(satz.Satz)
            segclass = segment.Segment
        elif string.lower(zeile[:5]) == "nach ":
            raise NotImplementedError
        else:
            raise Exception
        zeile = align.readline()
        while zeile != "":
            zeilenliste = string.split(zeile, "\t") # Zeile am Tab trennen
            if verbose:
                sys.stderr.write("Verarbeite Segment Nr. %s\n" %zeilenliste[0])
            slice1 = eval(zeilenliste[1])
            slice2 = eval(zeilenliste[2])
            if verbose:
                sys.stderr.write("%s zu %s\n" %(`slice1`, `slice2`))
            seg1 = segclass(l1[slice1[0]:slice1[-1]+1])
            seg2 = segclass(l2[slice2[0]:slice2[-1]+1])
            seg1.writeTagzeilen(out1.write)
            seg2.writeTagzeilen(out2.write)
            zeile = align.readline()
        align.close()
        out1.close()
        out2.close()
    dbconn.close()

def dokPaarUndAusgabe(zeile, ausgabeInsArbeitsverzeichnis):
    if   zeile[:14] == "Dokument-DB-ID":
        id = int(zeile[14:])
        dok1 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["de"])
        dok2 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["en"])
```

```
        out1 = open("db_"+‘id‘+".de.tag.al", "a")
        out2 = open("db_"+‘id‘+".en.tag.al", "a")
    elif zeile[:14] == "Dokument-Datei":
        name = string.strip(zeile[15:])
        dok1 = dokument.DokumentVonDatei(name+".de.tag")
        dok2 = dokument.DokumentVonDatei(name+".en.tag")
        if ausgabeInsArbeitsverzeichnis:
            name = os.path.basename(name)
        out1 = open(name+".de.tag.al", "a")
        out2 = open(name+".en.tag.al", "a")
    elif zeile[:6] == "Absatz":
        pos = string.find(zeile, "von")
        index = int(zeile[6:pos])
        temp = dokPaarUndAusgabe(zeile[pos+4:], ausgabeInsArbeitsverzeichnis)
        if not temp:
            sys.stderr.write("Dokument zu Absatz nicht angegeben\n")
            return None
        dok1 = temp[0].getRegionenMit(absatz.Absatz)[index]
        dok2 = temp[1].getRegionenMit(absatz.Absatz)[index]
        out1 = temp[2]
        out2 = temp[3]
    else:
        return None
    return (dok1, dok2, out1, out2)

if __name__ == ’__main__’:
    try:
        main()
    except:
        import traceback
        import StringIO
        tb = StringIO.StringIO()
        traceback.print_exc(file=tb)
        tb.seek(0)
        sys.stderr.write("in %s:\n" %sys.argv[0])
        zeile = tb.readline()
        while zeile != "":
            sys.stderr.write("\t"+zeile)
            zeile = tb.readline()
        dbconn.close()
        sys.exit(1)
```

M.3.4 align_1zu1.sh

```
#!/bin/bash

# align_1zu1.sh

# Aufruf: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en

ALIGN=‘dirname $0‘

if [ "$7" ] ; then

    test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
    test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
    test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
    test "$3" = "<dokende>"        && HDEL="dokument"
    test "$3" = "<DOKENDE>"        && HDEL="dokument"
    test "$3" = "<ABSATZ>"         && HDEL="absatz"
    test "$5" = "<ABSATZ>"         && SDEL="absatz"
```

```
    test "$5" = "<segmentgrenze>"  && SDEL="satz"
    test "$5" = "<SATZ>"           && SDEL="satz"
    test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
    test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }

    RED1=‘fgrep "<segmentgrenze>" "$6" | wc -l‘
    RED2=‘fgrep "<segmentgrenze>" "$7" | wc -l‘
    test "$RED1" = "$RED2" && echo "Warnung: Anzahl Segmente verschieden: $RED1 $RED2" >&2

    # Version, die 1:1 Alignment vornimmt
    # Achtung: Es wird nicht ueberprueft, ob die Anzahl der
    # Segmente ueberein stimmt.

    cp "$6" "$6.al"
    cp "$7" "$7.al"

else

    echo "Aufruf: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
    exit 1

fi
```

M.3.5 align_ai.sh

```
#!/bin/bash

# align.sh

# Aufruf: align.sh dokumentprefix abstandsmass
#   oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en

ALIGN=‘dirname $0‘

if [ "$7" ] ; then

    #MASS="TG2"
    MASS="WB2"
    #LEAVE_OUT_DIST=2.5
    #LEAVE_OUT_DIST_MAX=2.5
    #LEAVE_OUT_DIST_GRADIENT=0.00
    LEAVE_OUT_DIST=1.5
    LEAVE_OUT_DIST_MAX=3.53
    LEAVE_OUT_DIST_GRADIENT=0.12
    test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
    test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
    test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
    test "$3" = "<dokende>"        && HDEL="dokument"
    test "$3" = "<DOKENDE>"        && HDEL="dokument"
    test "$3" = "<ABSATZ>"         && HDEL="absatz"
    test "$5" = "<ABSATZ>"         && SDEL="absatz"
    test "$5" = "<segmentgrenze>"  && SDEL="satz"
    test "$5" = "<SATZ>"           && SDEL="satz"
    test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
    test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }

    RED1=‘dirname "$6"‘/‘basename "$6" .de.tag‘
    RED2=‘dirname "$7"‘/‘basename "$7" .en.tag‘
    STAMM="$6"
    test "$RED1" = "$RED2" && STAMM="$RED1"
```

```
36
37      # Version mit Abstandmatrix und -info Ausgabe
38
39      python "$ALIGN/nmatrix.py" -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_DIST
40      python "$ALIGN/mergemtr.py" "$STAMM".ai_pre "$STAMM".abstandinfo < "$STAMM".mtr_pre | tee "$STAMM".mtr |
41      java -jar "$ALIGN/al.jar" |
42      python "$ALIGN/align2al.py" - && rm "$STAMM".ai_pre "$STAMM".mtr_pre
43
44  elif [ "$2" ] ; then
45
46      python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/mergemtr
47
48  else
49
50      echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
51      echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
52      exit 1
53
54  fi
55
```

M.3.6 align_debug.sh

```
1   #!/bin/bash
2
3   # align.sh
4
5   # Aufruf: align.sh dokumentprefix abstandsmass
6   #   oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
7
8   ALIGN=‘dirname $0‘
9
10  if [ "$7" ] ; then
11
12      #MASS="TG2"
13      MASS="WB2"
14      #LEAVE_OUT_DIST=2.5
15      #LEAVE_OUT_DIST_MAX=2.5
16      #LEAVE_OUT_DIST_GRADIENT=0.00
17      LEAVE_OUT_DIST=1.5
18      LEAVE_OUT_DIST_MAX=3.53
19      LEAVE_OUT_DIST_GRADIENT=0.12
20      test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
21      test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
22      test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
23      test "$3" = "<dokende>"        && HDEL="dokument"
24      test "$3" = "<DOKENDE>"        && HDEL="dokument"
25      test "$3" = "<ABSATZ>"         && HDEL="absatz"
26      test "$5" = "<ABSATZ>"         && SDEL="absatz"
27      test "$5" = "<segmentgrenze>"  && SDEL="satz"
28      test "$5" = "<SATZ>"           && SDEL="satz"
29      test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
30      test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
31
32      RED1=‘dirname "$6"‘/‘basename "$6" .de.tag‘
33      RED2=‘dirname "$7"‘/‘basename "$7" .en.tag‘
34      STAMM="$6"
35      test "$RED1" = "$RED2" && STAMM="$RED1"
36
37      # Version mit vollstaendiger Ausgabe
38
```

```
    python "$ALIGN/nmatrix.py" -v -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OU
    python "$ALIGN/mergemtr.py" "$STAMM".ai_pre "$STAMM".abstandinfo < "$STAMM".mtr_pre | tee "$STAMM".mtr |
    java -jar "$ALIGN/al.jar" | tee "$STAMM".align |
    python "$ALIGN/align2al.py" -

elif [ "$2" ] ; then

    python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/merge

else

    echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
    echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
    exit 1

fi
```

M.3.7 align_mtr.sh

```
#!/bin/bash

# align_mtr.sh

# Aufruf: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en

ALIGN=‘dirname $0‘

if [ "$7" ] ; then

    test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
    test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
    test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
    test "$3" = "<dokende>"        && HDEL="dokument"
    test "$3" = "<DOKENDE>"        && HDEL="dokument"
    test "$3" = "<ABSATZ>"         && HDEL="absatz"
    test "$5" = "<ABSATZ>"         && SDEL="absatz"
    test "$5" = "<segmentgrenze>"  && SDEL="satz"
    test "$5" = "<SATZ>"           && SDEL="satz"
    test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
    test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }

    RED1=‘dirname "$6"‘/‘basename "$6" .de.tag‘
    RED2=‘dirname "$7"‘/‘basename "$7" .en.tag‘
    STAMM="$6"
    test "$RED1" = "$RED2" && STAMM="$RED1"

    # Version, die Abstandsmatrix .mtr voraussetzt

    test -f "$STAMM".mtr || { echo "Abstandsmatrix $STAMM.mtr nicht gefunden" >&2 ; exit 1 ; }

    java -jar -Xmx=650m "$ALIGN/al.jar" < "$STAMM".mtr | python "$ALIGN/align2al.py" -

else

    echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
    echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
    exit 1

fi
```

M.3.8 align_pur.sh

```
#!/bin/bash

# align.sh

# Aufruf: align.sh dokumentprefix abstandsmass
#   oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en

ALIGN=‘dirname $0‘

if [ "$7" ] ; then

    #MASS="TG2"
    MASS="WB2"
    #LEAVE_OUT_DIST=2.5
    #LEAVE_OUT_DIST_MAX=2.5
    #LEAVE_OUT_DIST_GRADIENT=0.00
    LEAVE_OUT_DIST=1.5
    LEAVE_OUT_DIST_MAX=3.53
    LEAVE_OUT_DIST_GRADIENT=0.12
    test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
    test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
    test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
    test "$3" = "<dokende>"        && HDEL="dokument"
    test "$3" = "<DOKENDE>"        && HDEL="dokument"
    test "$3" = "<ABSATZ>"         && HDEL="absatz"
    test "$5" = "<ABSATZ>"         && SDEL="absatz"
    test "$5" = "<segmentgrenze>"  && SDEL="satz"
    test "$5" = "<SATZ>"           && SDEL="satz"
    test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
    test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }

    # Version nur fuer .al

    python "$ALIGN/nmatrix.py" -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_DIST

elif [ "$2" ] ; then

    python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/mergemtr

else

    echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
    echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
    exit 1

fi

```

M.3.9 alignment.py

```
#!/usr/bin/python

# Kommandozeilentool alignment: List Jobanzahl + Abstandsmatrizen von stdin und
# gibt Ergebnis von Patricks MatrixAligner aus.

import popen2
import string
import sys

```

```
from DatabaseAPI import config

def main():
    anzahl = int(string.split(sys.stdin.readline())[0])
    sys.stderr.write("java -jar %s %s 2>/dev/null\n" %(config.al_jar, anzahl))
    r,w=popen2.popen2("java -jar %s %s 2>/dev/null" %(config.al_jar, anzahl))
    eingabe = sys.stdin.read()    # 2do: matrizenweise lesen und schreiben
    w.write(eingabe)              # oder besser threaded oder select IO
    w.close()
    ausgabe = r.read()
    r.close()
    sys.stdout.write(ausgabe)

if __name__ == '__main__':
    main()
```

M.3.10 alvis.py

```
#!/usr/bin/python

import os
import string
import sys
import types

from DatabaseAPI import dokumentdb, db, regiondb, config
from Region      import satz, dokument, segment, absatz

dbconn = db.Db()
regiondb.setDefaultDbconn(dbconn)

def main():
    global dbconn
    if len(sys.argv) == 2:
        align = open(sys.argv[1], "r")
    else:
        sys.stderr.write("Benutzung: %s <.align Datei>\n" %sys.argv[0])
        align = None
    if align:
        zeile = align.readline()
        if   zeile[:14] == "Dokument-DB-ID":
            id = int(zeile[14:])
            dok1 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["de"])
            dok2 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["en"])
            print "Dokument-DB-ID", id
        elif zeile[:14] == "Dokument-Datei":
            name = string.strip(zeile[15:])
            dok1 = dokument.DokumentVonDatei(name+".de.tag")
            dok2 = dokument.DokumentVonDatei(name+".en.tag")
            print "Dokument-Datei", name
        else:
            raise Exception
        zeile = align.readline()
        if   string.lower(zeile) == "nach satz\n":
            l1 = dok1.getRegionenMit(satz.Satz)
            l2 = dok2.getRegionenMit(satz.Satz)
            segclass = segment.Segment
        elif string.lower(zeile) == "nach absatz\n":
            l1 = dok1.getRegionenMit(absatz.Absatz)
            l2 = dok2.getRegionenMit(absatz.Absatz)
            segclass = dokument.Dokument
```

```
        else:
            raise Exception
        zeile = align.readline()
        while zeile != "":
            zeilenliste = string.split(zeile, "\t") # Zeile am Tab trennen
            segNr  = eval(zeilenliste[0])
            slice1 = eval(zeilenliste[1])
            slice2 = eval(zeilenliste[2])
            seg1 = segclass(l1[slice1[0]:slice1[-1]+1])
            seg2 = segclass(l2[slice2[0]:slice2[-1]+1])
            print "--- Segment Nr.", segNr, "Alignment", slice1, "-", slice2, "---"
            print seg1.getText()
            print seg2.getText()
            print
            zeile = align.readline()
        align.close()
    dbconn.close()

if __name__ == '__main__':
    main()
```

M.3.11 demo-a2.py

```
#!/usr/bin/env python

import GDK
import libglade
from gtk         import *
from Region      import region, dokument, segment, korpus, absatz
from DatabaseAPI import korpusdb, db
#import abstand

class Korpusbaumknoten(GtkTreeItem): #{

  def __init__(self, reg_de, reg_en, statusbar, matrix, sid = None):
    name = `reg_de` + ' / ' + `reg_en`
    GtkTreeItem.__init__(self, name)
    self.name = name
    self.status = statusbar
    self.reg_de = reg_de
    self.reg_en = reg_en
    if not sid:
      self.sid = statusbar.get_context_id('Korpus')
    else:
      self.sid = sid
    self.matrix = matrix
    self.exp_id = self.connect("expand", self.sig_expand)
    self.connect("select", self.sig_select)

  def is_leaf(self):
    return isinstance(self.reg_de, segment.Segment) \
        or isinstance(self.reg_en, segment.Segment)

  def init_subtree(self):
    if not self.is_leaf():
      self.subtree = GtkTree()
      self.subtree.set_selection_mode(SELECTION_BROWSE)
      self.set_subtree(self.subtree)
      self.subtree.show()

  def sig_select(self, data):
```

```
    if self.status:
      self.status.push(self.sid, self.name)
    if self.matrix:
      self.matrix.setRegionen(self.reg_de, self.reg_en)

  def sig_expand(self, _t):
    anzahl = len(self.reg_de)
    if anzahl > len(self.reg_en):
      anzahl = len(self.reg_en)
    for i in range(0, anzahl):
      item = Korpusbaumknoten(self.reg_de[i],
                              self.reg_en[i],
                              self.status, self.matrix, self.sid)
      self.subtree.append(item)
      item.init_subtree()
      item.show()
    self.disconnect(self.exp_id)
#}

class Abstandsmatrix: #{

  def __init__(self, drawingarea, window):
    self.drawingarea = drawingarea
    self.reg_de = region.Regionen([])   # leere Regionen
    self.reg_en = region.Regionen([])
    self.cellwidth  = 14
    self.cellheight = 14
    self.width  = 10
    self.height = 10
    self.xoff = 9
    self.yoff = 9
    self.window = window
    #self.abstand = abstand.AbstandChurchAndGale()
    #self.abstand = abstand.AbstandTest()

  def setRegionen(self, reg_de, reg_en):
    self.reg_de = reg_de
    self.reg_en = reg_en
    self.draw()

  def draw(self):
    self.drawingarea.show_now()

  def draw_old(self):
    reg_de = self.reg_de.getRegionenMit(segment.Segment)
    reg_en = self.reg_en.getRegionenMit(segment.Segment)
    x = len(reg_de)
    y = len(reg_de)
    if x > 0:
      x = 2 * self.xoff + self.cellwidth * (x-1)
    if y > 0:
      y = 2 * self.yoff + self.cellheight * (y-1)
    self.drawingarea.size(x, y)
    context  = self.drawingarea.get_window().new_gc()
    w = self.cellwidth
    h = self.cellheight
    for x in range(0, len(reg_de)):
      for y in range(0, len(reg_en)):
        #color = GdkColor(x+8*y,2*y+16*x,13*x+15*y,2)
        #context.foreground = color
        abstand = self.abstand.zwischen(reg_de[x], reg_en[y])
```

```
        print abstand, type(abstand)
        width   = 1 + self.width * abstand
        height  = 1 + self.height * abstand
        self.drawingarea.draw_rectangle(context, 1,
        self.xoff + w * x - width/2, self.yoff + h * y - height/2,
        width, height)
        # Idee: Abstandsmass = Groesse des Rechtecks
    self.drawingarea.show_now()
#}

class AbstandsmassDemoWindow(GtkWindow): #{

  def __init__(self, korpus_de, korpus_en):
    self.widgets = libglade.GladeXML('demo-a2.glade')
    zeichenflaeche = self.widgets.get_widget('drawingareaMatrix')
    fenster = self.widgets.get_widget('window1')
    matrix = Abstandsmatrix(zeichenflaeche, fenster)
    korpusbaum = self.widgets.get_widget('treeKorpus')
    statuszeile = self.widgets.get_widget('statusbar1')
    hauptknoten = Korpusbaumknoten(korpus_de, korpus_en, statuszeile, matrix)
    korpusbaum.append(hauptknoten)
    hauptknoten.init_subtree()
    hauptknoten.show()

  def connect(self, signal, function):
    fenster = self.widgets.get_widget('window1')
    fenster.signal_connect(signal, function)

#}

def main():
  #de = korpus.KorpusVonDateiliste(
  #       ["../wb/wbtag/wb2/sk50de.tag", "../wb/wbtag/wb2/sk500de.tag"])
  #en = korpus.KorpusVonDateiliste(
  #       ["../wb/wbtag/wb2/sk50en.tag", "../wb/wbtag/wb2/sk500en.tag"])
  dbconn = db.Db()
  de = korpusdb.KorpusVonDb(dbconn, "1")
  en = korpusdb.KorpusVonDb(dbconn, "2")
  win = AbstandsmassDemoWindow(de, en)
  win.connect("destroy", mainquit)
  win.connect("delete_event", mainquit)
  mainloop()
  dbconn.close()

if __name__ == '__main__':
  main()
```

M.3.12 demo-abstand.py

```
#!/usr/bin/python

libgladetest = 0

if libgladetest:
  import libglade

import popen2
import string
import sys
```

```
12 import tagger
13 import wb
14 import abstand
15 from DatabaseAPI import  config, db, regiondb
16 from Region      import  absatz
17
18 dbconn = db.Db()
19 regiondb.setDefaultDbconn(dbconn)
20
21 ab = { "C&G"   : abstand.AbstandChurchAndGale(),
22        "CG"    : abstand.AbstandChurchAndGale(),
23        "EINS"  : abstand.Abstand(1.0),
24        "TEST"  : abstand.AbstandTest(),
25        "NULL"  : abstand.Abstand(0.0),
26        "WB"    : abstand.AbstandWB(),
27        "WB2"   : abstand.AbstandWB_2(),
28        "WB2N"  : abstand.AbstandWB_2("de","en",1),    # nur Nomen
29        "TG"    : abstand.AbstandTG(),
30        "TG2"   : abstand.AbstandTG_2(),
31      }
32
33 wahl = "WB2"
34
35 sys.stdout.write(
36 """Bitte geben Sie zeilenweise abwechselnd deutschen und englischen Text ein.
37 Mit # kann das Abstandsmass gewechselt werden. ! wendet es auf das letzte
38 Sprachpaar an.
39 Mit +/- kann Abstandinfo ein- bzw. ausgeschaltet werden.
40 """)
41
42 sys.stdout.write("Deutsch:\n")
43 zeile = sys.stdin.readline()
44 de = 1
45 srcseg = None
46 dstseg = None
47 while zeile != "": #{
48   if len(zeile) > 0 and zeile[0] == "#": #{
49     zeile = zeile[1:sys.maxint]
50     zeile = string.strip(zeile)
51     if zeile == "":
52       sys.stdout.write("Aktuelles Abstandsmass %s: %s\n" %(‘wahl‘,‘ab[wahl]‘))
53       sys.stdout.write("Moegliche Abstandsmasse: " + ‘ab.keys()‘ + "\n")
54     elif ab.has_key(zeile):
55       wahl = zeile
56       sys.stdout.write("Aktuelles Abstandsmass %s: %s\n" %(‘wahl‘,‘ab[wahl]‘))
57     else:
58       sys.stdout.write("Abstandsmass " + ‘zeile‘ + " unbekannt\n")
59   #}
60   elif len(zeile) > 0 and zeile[0] == "+": #{
61     for key in ab.keys():
62       ab[key].setAiout(sys.stdout)
63     sys.stdout.write("Abstandinfo eingeschaltet.\n")
64   #}
65   elif len(zeile) > 0 and zeile[0] == "-": #{
66     for key in ab.keys():
67       ab[key].setAiout(abstand.NullFile())
68     sys.stdout.write("Abstandinfo ausgeschaltet.\n")
69   #}
70   elif len(zeile) > 0 and zeile[0] == "!": #{
71     if srcseg and dstseg:
72       wert = ab[wahl].zwischen(srcseg, dstseg)
```

```
      sys.stdout.write("Abstand: %s\n" %wert)
    else:
      sys.stdout.write("Es wurde weder Quell- noch Zielsegment eingegeben.\n")
  #}
  else: #{
    if de: #{
      sys.stdout.write("Tagging...\n")
      srcseg = tagger.tagline(zeile, "de")
      de = 0
    #}
    else: #{
      sys.stdout.write("Tagging...\n")
      dstseg = tagger.tagline(zeile, "en")
      wert = ab[wahl].zwischen(srcseg, dstseg)
      sys.stdout.write("Abstand:%s\n\n" %wert)
      de = 1
    #}
  #}
  if de:
    sys.stdout.write("Bitte deutschen Text eingeben:\n")
  else:
    sys.stdout.write("Bitte englischen Text eingeben:\n")
  zeile = sys.stdin.readline()
#}
```

M.3.13 demo-dok.py

```
#!/usr/bin/python

from Region import absatz
from Region import atom
from Region import dokument
from Region import satz
from Region import segment

print "Dokument de-test1.del:"
g = dokument.DokumentVonDatei("de-test1.del")
print "Text:", g
print "Struktur:", `g`
for klasse in [absatz.Absatz, segment.Segment, satz.Satz, atom.Atom]:
  r = g.getRegionenMit(klasse)
  print len(r), "Regionen zur Klasse", `klasse`
  for i in range(0, len(r)):
    print ("["+`i`+"]: " + `r[i]`)
  print "Wortliste:", r.getWoerter()
  print "Tagliste:", r.getTags()
  print "Lemmaliste:", r.getLemmata()
```

M.3.14 dokvis.py

```
#!/usr/bin/python

import os
import string
import sys
import types

from DatabaseAPI import dokumentdb, db, regiondb, config
from Region      import satz, dokument, segment, absatz, korpus

dbconn = db.Db()
```

```
regiondb.setDefaultDbconn(dbconn)

def main():
    global dbconn
    if len(sys.argv) >= 2 and sys.argv[1][:2] == "-r":
        reg = sys.argv[1][2:]
        del sys.argv[1]
        if not reg:
            reg = sys.argv[1]
            del sys.argv[1]
    else:
        reg = "satz"
    if len(sys.argv) == 2:
        dok = dokument.DokumentVonDatei(sys.argv[1])
    elif len(sys.argv) == 3:
        dbid = int(sys.argv[1])
        lang = config.lang2id[sys.argv[2]]
        dok = dokumentdb.DokumentVonDb(dbid, dbconn, lang)
    else:
        sys.stderr.write("Benutzung: %s [<Optionen>] <getaggtes Dokument>\n" %sys.argv[0])
        sys.stderr.write("oder     : %s [<Optionen>] <db-id> <sprache>\n" %sys.argv[0])
        sys.stderr.write("Optionen : -r Regionentyp  Werte: satz, segment, absatz\n")
        dok = None
    if dok:
        if   reg == "satz":
            l1 = dok.getRegionenMit(satz.Satz)
        elif   reg == "segment":
            l1 = dok.getRegionenMit(segment.Segment)
        elif   reg == "absatz":
            l1 = dok.getRegionenMit(absatz.Absatz)
        elif   reg == "dokument":
            l1 = korpus.Korpus([dok])
        else:
            raise Exception
        for index in range(len(l1)):
            print ("--- %s Nr." %reg), index, "---"
            print l1[index].getText()
            print
    dbconn.close()

if __name__ == '__main__':
    main()
```

M.3.15 eingabe.py

Siehe auch Seite 233 (Schnittstelle) und Seite 231 (Dateiformat).

```
#!/usr/bin/python

# Dieses Modull stellt eine Liste von zu bearbeitenden Dokumentpaaren
# bereit, die aus Kommandozeilenparametern gewonnen werden.

import sys
import StringIO
from Region import korpus, alignment

class Liste(korpus.KorpusMitSprache, alignment.AlignedRegion): #{

    def __init__(self, verbose = 0):
        self.setVerbose(verbose)
        dokumentliste = []
```

```
        sprache       = "undefiniert"
        korpus.KorpusMitSprache.__init__(self, dokumentliste, sprache)
        self.p_initOptions()
        self.closeOnExit = []
        if not regiondb.defaultDbconn:
            dbconn = db.Db()
            regiondb.setDefaultDbconn(dbconn)
            self.closeOnExit.append(dbconn)

    def close(self):
        for obj in self.closeOnExit:
            obj.close()
        self.closeOnExit = []

    def setVerbose(self, verbose):
        self.verbose = verbose

    def alignedRegion(self):
        liste = []
        for i in self:
            liste.append(i.alignedRegion())
        return korpus.KorpusMitSprache(liste, self.p_zs)  # 2do: auch aligned

    def getQuellsprache(self):
        return self.p_qs

    def getZielsprache(self):
        return self.p_zs

    def p_initOptions(self):
        self.options_l = {
            'c' : "string    use string as input line",
            'd' : "language  set destination language",
            'i' : "filename  read all input from file",
            'j' : "filename  read dlang-input from file",
            's' : "language  set source language",
        }
        self.options_s = {
            '.' : "          dump current '-c' buffer",
            'l' : "          input is a list of input filenames",
            't' : "          tag input and treat lines as paragraphs",
        }
        self.p_fin1     = None
        self.p_fin2     = None
        self.p_fname_qs = None
        self.p_fname_zs = None
        self.p_islist   = 0
        self.p_tag      = 0
        self.p_qs       = "de"
        self.p_zs       = "en"
        self.p_cmdinput = StringIO.StringIO()

    def eatOptions(self, argv):
        state = 0
        error = 0
        verbose = self.verbose
        index = 1
        laenge = len(argv)
        while index < len(argv): #{
            arg = argv[index]
            if not state:
```

```
                for i in self.options_l.keys():
                    if arg[:2] == "-"+i:
                        state = i
                        if arg[2:] == "":
                            del argv[index]
                        else:
                            argv[index] = arg[2:]
                        break
                for i in self.options_s.keys():
                    if arg[:2] == "-"+i:
                        state = i
                        if arg[2:] != "":
                            if verbose:
                                sys.stderr.write("bad argument %s\n"%‘arg‘)
                            error = 1
                        break
                if not state:
                    break;       # leave eatOptions()
                continue
            if state == ’.’:
                if verbose:
                    sys.stderr.write("dumping StringIO...\n")
                cmdinput.seek(0)
                sys.stdout.write(self.p_cmdinput.read())
            elif state == ’i’:
                if arg == "-":
                    self.p_fin1 = sys.stdin
                    if verbose:
                        sys.stderr.write("input1&2 set to stdin\n")
                else:
                    self.p_fname_qs = arg
                    self.p_fin1 = open(arg, "r")
                    if verbose:
                        sys.stderr.write("input1&2 set to file %s\n" %arg)
                self.p_fin2 = self.p_fin1
            elif state == ’c’:
                self.p_cmdinput.write(arg+’\n’)
                if verbose:
                    sys.stderr.write("StringIO.write(%s)\n"%‘arg+’\n’‘)
            elif state == ’s’:
                self.p_qs = arg
                if verbose:
                    sys.stderr.write("source language = %s\n" %arg)
            elif state == ’t’:
                self.p_tag = 1
                if verbose:
                    sys.stderr.write("tag = 1\n")
            elif state == ’l’:
                self.p_islist = 1
                if verbose:
                    sys.stderr.write("islist = 1\n")
            elif state == ’d’:
                self.p_zs = arg
                if verbose:
                    sys.stderr.write("destination language = %s\n" %arg)
            elif state == ’j’:
                if arg == "-":
                    self.p_fin2 = sys.stdin
                    if verbose:
                        sys.stderr.write("input2 set to stdin\n")
                else:
```

```
                self.p_fname_zs = arg
                self.p_fin2 = open(arg, "r")
                if verbose:
                    sys.stderr.write("input2 set to file %s\n" %arg)
            else:
                raise NotImplementedError
            del argv[index]
            state = 0
        #}
        return laenge - len(argv)

    def finishOptions(self):
        verbose = self.verbose
        if self.p_cmdinput.tell() != 0:
            self.p_cmdinput.seek(0)
            fin1 = self.p_cmdinput
            fin2 = fin1
            if verbose:
                sys.stderr.write("input from '-c' buffer\n")
            fname_qs = None
            fname_zs = None
        else:
            fname_qs = self.p_fname_qs
            fname_zs = self.p_fname_zs
            fin1 = self.p_fin1
            fin2 = self.p_fin2
        # create list of DokumentPaar objects according to islist and tag
        if verbose and fin1 == sys.stdin:
            sys.stderr.write("expecting input from stdin\n")
        dokumentPaare = []
        qs = self.p_qs
        zs = self.p_zs
        if fin1 and self.p_islist:
            fnamedic = {}
            fname_qs = fin1.readline()
            fname_zs = fin2.readline()
            while fname_qs and fname_zs:
                fname_qs = fname_qs[:-1]     # Newline abschneiden
                fname_zs = fname_zs[:-1]
                if not self.p_tag:
                    dp = dokument.DokumentPaarVonDateien(fname_qs, fname_zs,
                                                         qs, zs)
                else:
                    tfin1 = open(fname_qs, "r")
                    tfin2 = open(fname_zs, "r")
                    dp = dokument.DokumentPaarVonRohtexte(tfin1, tfin2, qs, zs)
                    tfin1.close()
                    tfin2.close()
                dokumentPaare.append(dp)
                fname_qs = fin1.readline()
                fname_zs = fin2.readline()
        elif fin1:
            if fname_qs and fname_zs:
                dp = dokument.DokumentPaarVonDateien(fname_qs, fname_zs, qs,zs)
            elif not self.p_tag:
                dp = dokument.DokumentPaarVonStreams(fin1, fin2, qs, zs)
            else:
                dp = dokument.DokumentPaarVonRohtexte(fin1, fin2, qs, zs)
            dokumentPaare.append(dp)
        if fin1:
            fin1.close()
```

```
            if fin1 != fin2:
                fin2.close()
        self.p_setDokumentliste(dokumentPaare)
        self.p_setSprache(qs)

    def help(self, file):
        keys = self.options_l.keys() + self.options_s.keys()
        keys.sort()
        for key in keys:
            if self.options_l.has_key(key):
                file.write("          -%s %s\n" %(key, self.options_l[key]))
            elif self.options_s.has_key(key):
                file.write("          -%s %s\n" %(key, self.options_s[key]))
            else:
                raise ValueError

#}

from DatabaseAPI import db, regiondb

#
# getRegion() liefert RegionenMit-Objekt zum als String gegebenen Regionentyp
#

def getRegionen(dok, reg):
    if   reg == "satz":
        l1 = dok.getRegionenMit(satz.Satz)
    elif   reg == "segment":
        l1 = dok.getRegionenMit(segment.Segment)
    elif   reg == "absatz":
        l1 = dok.getRegionenMit(absatz.Absatz)
    elif   reg == "dokument":
        l1 = dok.getRegionenMit(dokument.Dokument)
    elif   reg == "korpus":
        l1 = dok.getRegionenMit(korpus.Korpus)
    else:
        raise Exception
    return l1

##########################################################################
# Test-Skript

from Region import korpus, dokument, absatz, segment, satz

def main():
    nebeneinander = 0
    verbose = 0
    hilfe = 0
    reg = "satz"
    dok = Liste(verbose)
    while len(sys.argv) > 1: #{
        #if verbose:
        #    sys.stderr.write("args left: %s\n" %‘sys.argv[1:]‘)
        if sys.argv[1] == "-h":
            hilfe = 1
            break
        elif sys.argv[1] == "-v":
            verbose = 1
            dok.setVerbose(verbose)
            del sys.argv[1]
        elif sys.argv[1] == "-n":
```

```
            nebeneinander = 1
            del sys.argv[1]
        elif sys.argv[1][:2] == "-r":
            reg = sys.argv[1][2:]
            del sys.argv[1]
            if not reg:
                try:
                    reg = sys.argv[1]
                    del sys.argv[1]
                except:
                    hilfe = 1
                    break
        else:
            eaten = dok.eatOptions(sys.argv)
            if not eaten:
                hilfe = 1
                break
            if verbose:
                sys.stderr.write("eingabe.Liste has eaten %s args\n" %eaten)
    if hilfe:
        if verbose:
            sys.stderr.write("args left: %s\n" %`sys.argv[1:]`)
        sys.stderr.write("Usage: %s <options>\n" %sys.argv[0])
        sys.stderr.write("options:  -h           prints this help\n")
        sys.stderr.write("          -n           alternate docs in output\n")
        sys.stderr.write("          -r regtype   soft regions: one of\n")
        for i in ["satz", "segment", "absatz", "dokument"]:
            sys.stderr.write(27*' ' + "%s\n" %i)
        sys.stderr.write("          -v           verbose\n")
        dok.help(sys.stderr)
        dok.close()
        sys.exit(0)
    if verbose:
        sys.stderr.write("reading input...\n")
    dok.finishOptions()
    if verbose:
        sys.stderr.write("region type = %s\n" %reg)
    l1 = getRegionen(dok, reg)
    l2 = getRegionen(dok.alignedRegion(), reg)
    if verbose:
        sys.stderr.write("number of regions: %s\n" %`(len(l1),len(l2))`)
    if nebeneinander:
        for index in range(max(len(l1), len(l2))):
            print ("--- %s Nr." %reg), index, "---"
            if l1.has_key(index):
                print l1[index].getText()
            if l2.has_key(index):
                print l2[index].getText()
            print
    else:
        for (header, l) in [("Quellsprache", l1), ("Zielsprache", l2)]:
            print "### %s ###" %header
            print
            for index in range(len(l)):
                print ("--- %s Nr." %reg), index, "---"
                print l[index].getText()
                print
    dok.close()

if __name__ == '__main__':
    main()
```

M.3.16 getalign.py

```
#!/usr/bin/python

import abstand
import os
import sys
import types

from DatabaseAPI import dokumentdb, db, regiondb, config
from Region      import satz, dokument, segment

dbconn = db.Db()
regiondb.setDefaultDbconn(dbconn)

blockgroesse = 1
if blockgroesse != 1:
  print "Derzeit nur Bloeckgroesse 1 unterstuetzt."
  sys.exit(1)

def usage():
    print "Usage: getalign.py -d <DB-IDs> | -s <basename>"
    sys.exit()

def main():
    global dbconn
    if len(sys.argv) < 2:
        usage()
    if sys.argv[1] == "-d":
        liste = map(lambda x: int(x), sys.argv[2:])
    elif sys.argv[1] == "-s":
        liste = sys.argv[2:]
    else:
        usage()
    for name in liste:
        if type(name) == types.StringType:
            fname = name
        elif type(name) == types.IntType:
            fname = 'db_'+`name`
        else:
            raise TypeError
        align = open(os.path.basename(fname)+".align", "w")
        if type(name) == types.StringType:
            dok1 = dokument.DokumentVonDatei(name+".de.tag.al")
            dok2 = dokument.DokumentVonDatei(name+".en.tag.al")
            align.write("Dokument-Datei "+name+"\n")
        elif type(name) == types.IntType:
            dok1 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["de"])
            dok2 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["en"])
            align.write("Dokument-DB-ID "+`name`+"\n")
        else:
            raise TypeError
        l1 = dok1.getRegionenMit(segment.Segment)
        l2 = dok2.getRegionenMit(segment.Segment)
        align.write("nach Satz\n")
        s1 = 0
        s2 = 0
        laenge = len(l1)
        if len(l2) < laenge:
            laenge = len(l2)
            sys.stderr.write("Warnung: " + name + " Anzahl Seg. verschieden\n")
```

```
        for i in range(0,laenge):
            s1n = s1+len(l1[i])
            s2n = s2+len(l2[i])
            align.write('i'             + \
            "\t" +      'range(s1,s1n)' + \
            "\t" +      'range(s2,s2n)' + "\n")
            s1 = s1n
            s2 = s2n
        align.close()
    dbconn.close()

if __name__ == '__main__':
    main()
```

M.3.17 matrix.py

```
#!/usr/bin/python

import abstand
import os
import sys
import types

from DatabaseAPI import dokumentdb, db, regiondb, config
from Region      import satz, dokument, segment

dbconn = db.Db()
regiondb.setDefaultDbconn(dbconn)

blockgroesse = 1
if blockgroesse != 1:
  sys.stderr.write("Derzeit nur Bloeckgroesse 1 unterstuetzt."+'\n')
  sys.exit(1)

def main():
    global dbconn
    jobs={}
    jobs["default"] = {}
    #jobs["default"]["abstand"] = abstand.Abstand()
    #jobs["default"]["abstand"] = abstand.AbstandTest()
    #jobs["default"]["abstand"] = abstand.AbstandChurchAndGale()
    #jobs["default"]["abstand"] = abstand.AbstandTG()
    #jobs["default"]["abstand"] = abstand.AbstandWB()
    jobs["default"]["abstand"] = abstand.AbstandWB_2()
    #jobs["default"]["abstand"] = abstand.AbstandWB_2("de","en",1) # nurNomen
    for mass in jobs.keys():
        jobs[mass]["file"] = sys.stdout
    for name in sys.argv[1:]:
        if type(name) == types.StringType:
            fname = name
        elif type(name) == types.IntType:
            fname = 'db_'+'name'
        else:
            raise TypeError
        if type(name) == types.StringType:
            dok1 = dokument.DokumentVonDatei(name+".de.tag")
            dok2 = dokument.DokumentVonDatei(name+".en.tag")
            for mass in jobs.keys():
                jobs[mass]["file"].write("Dokument-Datei "+name+"\n")
        elif type(name) == types.IntType:
            dok1 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["de"])
```

```
            dok2 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["en"])
            for mass in jobs.keys():
                jobs[mass]["file"].write("Dokument-DB-ID "+‘name‘+"\n")
        else:
            raise TypeError
        l1 = dok1.getRegionenMit(segment.Segment)
        l2 = dok2.getRegionenMit(segment.Segment)
        s1 = 0
        s2 = 0
        laenge = len(l1)
        if len(l2) < laenge:
            laenge = len(l2)
            sys.stderr.write("Warnung: " + name + " Anzahl Seg. verschieden\n")
        for i in range(0,laenge):
            s1n = s1+len(l1[i])
            s2n = s2+len(l2[i])
            s1 = s1n
            s2 = s2n
        l1 = dok1.getRegionenMit(satz.Satz)
        l2 = dok2.getRegionenMit(satz.Satz)
        for job in jobs.values():
            job["file"].write("nach Satz\n")
            job["file"].write(‘len(l2)-blockgroesse+1‘ + "\t")
            job["file"].write(‘len(l1)-blockgroesse+1‘ + "\n")
        for y in range(0, len(l1)-blockgroesse+1):
            for x in range(0, len(l2)-blockgroesse+1):
                # sys.stderr.write(’\n’)
                # sys.stderr.write(("--- %s - %s ---" %(‘y‘,‘x‘))+’\n’)
                # sys.stderr.write(("DE: %s" %l1[y].getText())+’\n’)
                # sys.stderr.write(("EN: %s" %l2[x].getText())+’\n’)
                # sys.stderr.write(’\n’)
                abstaende=[]
                for job in jobs.values():
                    a = job["abstand"].zwischen(l1[y], l2[x])
                    job["file"].write(‘a‘ + "\n")
                    abstaende.append(a)
                # sys.stderr.write("AB: %s" %‘abstaende‘)
        # sys.stderr.write(‘name‘ + " erledigt.\n")
    jobs["default"]["abstand"].close()
    dbconn.close()

def dateiliste():
  return [
’../korpus/de-news/1997/05/de-news-1997-05-30’,
403,
’../korpus/de-news/1997/05/de-news-1997-05-19’,
’../korpus/de-news/1997/05/de-news-1997-05-03’,
’../korpus/de-news/1997/05/de-news-1997-05-11’,
’../korpus/de-news/1997/05/de-news-1997-05-06’,
]

dummyliste = [
’../korpus/de-news/1997/05/de-news-1997-05-01’,
’../korpus/de-news/1997/05/de-news-1997-05-02’,
’../korpus/de-news/1997/05/de-news-1997-05-03’,
’../korpus/de-news/1997/05/de-news-1997-05-04’,
’../korpus/de-news/1997/05/de-news-1997-05-05’,
’../korpus/de-news/1997/05/de-news-1997-05-06’,
’../korpus/de-news/1997/05/de-news-1997-05-07’,
’../korpus/de-news/1997/05/de-news-1997-05-08’,
’../korpus/de-news/1997/05/de-news-1997-05-09’,
```

```
'../korpus/de-news/1997/05/de-news-1997-05-10',
'../korpus/de-news/1997/05/de-news-1997-05-11',
'../korpus/de-news/1997/05/de-news-1997-05-12',
'../korpus/de-news/1997/05/de-news-1997-05-13',
'../korpus/de-news/1997/05/de-news-1997-05-14',
'../korpus/de-news/1997/05/de-news-1997-05-15',
'../korpus/de-news/1997/05/de-news-1997-05-16',
'../korpus/de-news/1997/05/de-news-1997-05-17',
'../korpus/de-news/1997/05/de-news-1997-05-19',
'../korpus/de-news/1997/05/de-news-1997-05-20',
'../korpus/de-news/1997/05/de-news-1997-05-21',
'../korpus/de-news/1997/05/de-news-1997-05-22',
'../korpus/de-news/1997/05/de-news-1997-05-23',
'../korpus/de-news/1997/05/de-news-1997-05-24',
'../korpus/de-news/1997/05/de-news-1997-05-26',
'../korpus/de-news/1997/05/de-news-1997-05-27',
'../korpus/de-news/1997/05/de-news-1997-05-28',
'../korpus/de-news/1997/05/de-news-1997-05-30',
'../korpus/de-news/1997/05/de-news-1997-05-31',
]

if __name__ == '__main__':
    main()
```

M.3.18 matrix.sh

```
#!/bin/bash

# matrix.sh erstellt eine fuer ein Dokumentpaar
#     1. absatzweise Abstandsmatrizen in $1_$2.abs.mtr
#     2. Gesamtabstandsmatrix         in $1_$2.mtr
#     3. Fehler- und Diagnoseausgabe  in temp.out
# und fuer WB2 zusaetzlich
#     4. absatzweise Abstandinfo      in $I.abs.ai
#     5. Gesamtabstandinfo            in $I.abstandinfo

# Aufruf: matrix.sh dokumentprefix abstandsmass

ALIGN=‘dirname $0‘

test "$2" = WB2 && python "$ALIGN/nmatrix.py" -v -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" -o "$1_

test "$2" != WB2 && python "$ALIGN/nmatrix.py" -v -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" -o "$1
```

M.3.19 mavis.sh

```
#!/bin/bash

#JAVA=/usr/lib/jdk1.3/bin/java
JAVA=java

$JAVA de.denkselbst.mavis.Mavis $1
```

M.3.20 mergemtr.py

```
#!/usr/bin/python


import string
import sys

```

```
aiin  = None   # Anstandinfo
aiout = None
if len(sys.argv) > 1:
    aiin  = open(sys.argv[1], "r")
if len(sys.argv) > 2:
    aiout = open(sys.argv[2], "w")

state = "header"
breiten   = []
hoehen    = []
matrizen  = []
abstandinfos  = []

def fehler(ueberschrift):
    sys.stderr.write("%s\n" %ueberschrift)
    sys.stderr.write("aktuelle Zeile: zeile = %s\n" %‘zeile‘)
    sys.stderr.write("bisher gelesen: daten = %s\n" %‘daten‘)
    sys.stderr.write("bisher gelesen: len   = %s\n" %‘len(daten)‘)
    sys.stderr.write("notwendig: breite*hoehe = %s\n" %‘breite*hoehe‘)
    sys.exit(1)

header_ausgegeben = 0
zeile = sys.stdin.readline()
while zeile != "":
    if state == "header":
        if zeile[:len("Absatz ")] == "Absatz ":
            if not header_ausgegeben:
                sys.stdout.write(zeile[string.find(zeile, " von ")+5:])
            state = "header2"
        elif zeile[-len(" jobs follow\n"):] == " jobs follow\n":
            state = "header"
        else:
            sys.stderr.write("Quelleninfo %s nicht verstanden\n" %‘zeile‘)
            raise Exception
    elif state == "header2":
        if string.lower(zeile) == "nach satz\n":
            if not header_ausgegeben:
                sys.stdout.write(zeile)
                header_ausgegeben = 1
            state = "header3"
        else:
            sys.stderr.write("Softregiontyp %s nicht verstanden\n" %‘zeile‘)
            raise Exception
    elif state == "header3":
        liste = string.split(zeile, ’\t’)
        if len(liste) == 2:
            try:
                breite = int(liste[0])
                hoehe  = int(liste[1])
                daten  = []
                aidaten  = []
                state  = "daten"
                #sys.stderr.write("Teilmatrix %s\n" %‘(breite,hoehe)‘)
                if len(daten) == breite * hoehe:
                    breiten.append(breite)
                    hoehen.append(hoehe)
                    matrizen.append(daten)
                    abstandinfos.append(aidaten)
                    state = "header"
            except ValueError:
                pass
```

```
    elif state == "daten":
        liste = string.split(zeile)
        try:
            for datum in liste:
                daten.append(float(datum))
                aidatum = []
                if aiin:
                    aizeile = aiin.readline()
                    while aizeile != "<SATZ>\n":
                        aidatum.append(string.rstrip(aizeile))
                        if aizeile == "":
                            fehler("EOF in abstandinfos zu frueh")
                            raise Exception     # EOF zu frueh
                        aizeile = aiin.readline()
                    aidatum.append("")    # fuer abschliessendes Newline
                aidaten.append(string.join(aidatum, '\n'))
        except ValueError:
            fehler("Fehlerhafte Eingabe")
        if len(daten) >= breite * hoehe:
            breiten.append(breite)
            hoehen.append(hoehe)
            matrizen.append(daten)
            abstandinfos.append(aidaten)
            state = "header"
    zeile = sys.stdin.readline()
breite = 0
for i in breiten:
    breite = breite + i
hoehe = 0
for i in hoehen:
    hoehe = hoehe + i
sys.stdout.write("%s\t%s\n" %(breite, hoehe))
if aiout:
    aiout.write("%s\t%s\n" %(breite, hoehe))
prefix = 0
suffix = breite
for mindex in range(len(matrizen)):
    mhoehe  = hoehen[mindex]
    mbreite = breiten[mindex]
    matrix  = matrizen[mindex]
    abstandinfo  = abstandinfos[mindex]
    suffix = suffix - mbreite
    for my in range(mhoehe):
        sys.stdout.write(prefix * "1.0\n")
        if aiout:
            aiout.write(prefix * "nicht berechnet, da ausserhalb des Absatzalignments\n<SATZ>\n")
        for mx in range(mbreite):
            sys.stdout.write("%s\n" %matrix[mx+my*mbreite])
            if aiout:
                aiout.write(abstandinfo[mx+my*mbreite])
                aiout.write("<SATZ>\n")
        sys.stdout.write(suffix * "1.0\n")
        if aiout:
            aiout.write(suffix * "nicht berechnet, da ausserhalb des Absatzalignments\n<SATZ>\n")
    prefix = prefix + mbreite
```

M.3.21 mtr2mavis.py

```
#!/usr/bin/python

# mtr2mavis.py
# passt die Abstandsmatrizen fuer Dateien, die nicht dem Schema
# $STAMM.de.tag / $STAMM.en.tag entsprechen, an und legt Links
# an, um nach diesem Schema auf die Dateien zugreifen zu koennen.

import os
import string
import sys

if len(sys.argv) <= 1:
    sys.stderr.write("Benutzung: %s <Liste von .mtr Dateien>\n" %sys.argv[0])
    sys.exit(1)

def neueNamen(name1, name2):
    ldp = min([len(name1), len(name2)])
    while name1[:ldp] != name2[:ldp]:
        ldp = ldp - 1
    lfdnr = 1
    while 1:
        stamm = name1[:ldp] + "~" + `lfdnr`
        name_de = stamm + ".de.tag"
        name_en = stamm + ".en.tag"
        name_mtr = stamm + ".mtr"
        if  not os.access(name_de, os.F_OK) \
        and not os.access(name_en, os.F_OK) \
        and not os.access(name_mtr, os.F_OK):
            break
        lfdnr = lfdnr + 1
    return (name_de, name_en, name_mtr, stamm)

for mtrfilename in sys.argv[1:]:

    mtrfile = open(mtrfilename, "r")
    zeile = mtrfile.readline()
    if zeile[:len("Dokument-Datei ")] == "Dokument-Datei ":
        sys.stderr.write("%s bereits im richtigem Format\n" %mtrfilename)
        mtrfile.close()
        continue
    elif zeile[-len(" jobs follow\n"):] == " jobs follow\n":
        sys.stderr.write("%s erfordert zuerst mergemtr.py \n" %mtrfilename)
        mtrfile.close()
        continue
    elif zeile[:len("Paar Dokument-Datei ")] == "Paar Dokument-Datei ":
        pos = string.find(zeile, " und Dokument-Datei")
        name1 = string.strip(zeile[20:pos])
        name2 = string.strip(zeile[pos+20:])
        name_de, name_en, name_mtr, stamm = neueNamen(name1, name2)
        mtrneu = open(name_mtr, "w")
        mtrneu.write("Dokument-Datei %s\n" %stamm)  # Header neu
        mtrneu.write(mtrfile.read())                # Rest kopieren
        mtrneu.close()
        mtrfile.close()
        os.link(name1, name_de)
        os.link(name2, name_en)
        name_ai = name1 + ".abstandinfo"
        if os.access(name_ai, os.F_OK):
            os.link(name_ai, stamm + ".abstandinfo")
```

```
    else:
        sys.stderr.write("%s nicht behandelt, Header unbekannt\n" %mtrfilename)
        mtrfile.close()

```

M.3.22 nmatrix.py

```
#!/usr/bin/python

# Kommandozeilentool matrix.py

import math
import sys

import abstand
import eingabe
from DatabaseAPI import  config, db, regiondb
from Region      import  atom, absatz, region

def main():
    dbconn = db.Db()
    regiondb.setDefaultDbconn(dbconn)
    ab = { "C&G"   : abstand.AbstandChurchAndGale(),
           "CG"    : abstand.AbstandChurchAndGale(),
           "EINS"  : abstand.Abstand(1.01),
           "TEST"  : abstand.AbstandTest(),
           "NULL"  : abstand.Abstand(0.0),
           "WB"    : abstand.AbstandWB(),
           "WB2"   : abstand.AbstandWB_2(),
           "WB2N"  : abstand.AbstandWB_2("de","en",1),    # nur Nomen
           "TG"    : abstand.AbstandTG(),
           "TG2"   : abstand.AbstandTG_2(),
         }
    wahl = "TG"
    fout = sys.stdout
    aiout = None
    verbose = 0
    text = 0
    leave_out = 0
    leave_out_dist_gradient = 0.0
    leave_out_dist_max   = -1.0
    hilfe = len(sys.argv) == 1
    reg_soft = "segment"
    reg_hard = "dokument"
    korp = eingabe.Liste(verbose)
    while len(sys.argv) > 1: #{
        #if verbose:
        #    sys.stderr.write("args left: %s\n" %‘sys.argv[1:]‘)
        if sys.argv[1] == "-h":
            hilfe = 1
            break
        elif sys.argv[1] == "-u":
            text = 1
            del sys.argv[1]
        elif sys.argv[1] == "-v":
            verbose = 1
            korp.setVerbose(verbose)
            del sys.argv[1]
        elif sys.argv[1][:2] == "-l":
            leave_out_dist = sys.argv[1][2:]
            del sys.argv[1]
```

```
55              if not leave_out_dist:
56                  try:
57                      leave_out_dist = sys.argv[1]
58                      del sys.argv[1]
59                  except:
60                      hilfe = 1
61                      break
62              try:
63                  leave_out_dist = float(leave_out_dist)
64              except ValueError:
65                  hilfe = 1
66                  break
67              leave_out = 1
68              if verbose:
69                  sys.stderr.write("cells more than %s units " %leave_out_dist)
70                  sys.stderr.write("from the conectiong line will be left out\n")
71          elif sys.argv[1][:2] == "-L":
72              leave_out_dist_gradient = sys.argv[1][2:]
73              del sys.argv[1]
74              if not leave_out_dist_gradient:
75                  try:
76                      leave_out_dist_gradient = sys.argv[1]
77                      del sys.argv[1]
78                  except:
79                      hilfe = 1
80                      break
81              try:
82                  leave_out_dist_gradient = float(leave_out_dist_gradient)
83              except ValueError:
84                  hilfe = 1
85                  break
86              if verbose:
87                  sys.stderr.write("The leave out distance will be increased by ")
88                  sys.stderr.write("%s units for each " %leave_out_dist_gradient)
89                  sys.stderr.write("unit along the\n connecting line (upper left")
90                  sys.stderr.write(" to lower right) up to the middle of each ")
91                  sys.stderr.write("hard-region\n")
92          elif sys.argv[1][:2] == "-M":
93              leave_out_dist_max = sys.argv[1][2:]
94              del sys.argv[1]
95              if not leave_out_dist_max:
96                  try:
97                      leave_out_dist_max = sys.argv[1]
98                      del sys.argv[1]
99                  except:
100                     hilfe = 1
101                     break
102             try:
103                 leave_out_dist_max = float(leave_out_dist_max)
104             except ValueError:
105                 hilfe = 1
106                 break
107             if verbose:
108                 sys.stderr.write("If -l is used, leave out distance will not ")
109                 sys.stderr.write("exceed %s units.\n" %leave_out_dist_max)
110         elif sys.argv[1][:2] == "-r":
111             reg_soft = sys.argv[1][2:]
112             del sys.argv[1]
113             if not reg_soft:
114                 try:
115                     reg_soft = sys.argv[1]
```

```
            del sys.argv[1]
        except:
            hilfe = 1
            break
    if verbose:
        sys.stderr.write("soft region type = %s\n" %reg_soft)
elif sys.argv[1][:2] == "-R":
    reg_hard = sys.argv[1][2:]
    del sys.argv[1]
    if not reg_hard:
        try:
            reg_hard = sys.argv[1]
            del sys.argv[1]
        except:
            hilfe = 1
            break
    if verbose:
        sys.stderr.write("hard region type = %s\n" %reg_hard)
elif sys.argv[1][:2] == "-m":
    wahl = sys.argv[1][2:]
    del sys.argv[1]
    if not wahl:
        try:
            wahl = sys.argv[1]
            del sys.argv[1]
        except:
            hilfe = 1
            break
    if verbose:
        sys.stderr.write("distance measure = %s\n" %wahl)
elif sys.argv[1][:2] == "-o":
    fname = sys.argv[1][2:]
    del sys.argv[1]
    if not fname:
        try:
            fname = sys.argv[1]
            del sys.argv[1]
        except:
            hilfe = 1
            break
    fout = open(fname, "w")
    if verbose:
        sys.stderr.write("output to file %s\n" %fname)
elif sys.argv[1][:2] == "-O":
    fname = sys.argv[1][2:]
    del sys.argv[1]
    if not fname:
        try:
            fname = sys.argv[1]
            del sys.argv[1]
        except:
            hilfe = 1
            break
    aiout = open(fname, "w")
    if verbose:
        sys.stderr.write("abstandinfo %s\n" %fname)
else:
    eaten = korp.eatOptions(sys.argv)
    if not eaten:
        hilfe = 1
        break
```

```
177             if verbose:
178                 sys.stderr.write("eingabe.Liste has eaten %s args\n" %eaten)
179         if hilfe:
180             if verbose:
181                 sys.stderr.write("args left: %s\n" %‘sys.argv[1:]‘)
182             sys.stderr.write("Usage: %s <options>\n" %sys.argv[0])
183             sys.stderr.write("options:  -h           print out this help\n")
184             sys.stderr.write("          -l n         leave out cell which are more than n units away\n")
185             sys.stderr.write("                       from the conectiong line\n")
186             sys.stderr.write("          -L gradient  modifies leave out distance\n")
187             sys.stderr.write("          -M maximum   modifies leave out distance\n")
188             sys.stderr.write("          -m distmeas  choose distance measure:\n")
189             keys = ab.keys()
190             keys.sort()
191             for w in keys:
192                 sys.stderr.write(25*’ ’ + "%s\t%s\n" %(w,‘ab[w]‘))
193             sys.stderr.write("          -o filename  write output to file\n")
194             sys.stderr.write("          -O filename  write abstandinfo to file\n")
195             sys.stderr.write("          -r regtype   soft regions: one of\n")
196             for i in ["satz", "segment", "absatz", "dokument"]:
197                 sys.stderr.write(25*’ ’ + "%s\n" %i)
198             sys.stderr.write("          -R regtype   hard regions: one of\n")
199             for i in ["segment", "absatz", "dokument", "korpus"]:
200                 sys.stderr.write(25*’ ’ + "%s\n" %i)
201             sys.stderr.write("          -u           output index and text\n")
202             sys.stderr.write("          -v           verbose mode\n")
203             korp.help(sys.stderr)
204             korp.close()
205             for key in ab.keys():
206                 ab[key].close()
207             dbconn.close()
208             sys.exit(1)
209         if verbose:
210             sys.stderr.write("distance measure %s = %s\n" %(wahl,‘ab[wahl]‘))
211         if aiout:
212             for key in ab.keys():
213                 ab[key].setAiout(aiout)   # Abstandinfo
214         korp.finishOptions()
215         hardregions = eingabe.getRegionen(korp, reg_hard)
216         fout.write("%s jobs follow\n" %len(hardregions))
217         try:
218             dok = hardregions[len(hardregions)-1]
219             alignedDok = dok.alignedRegion()
220         except AttributeError:
221             sys.stderr.write("Error: no aligned hardregion for last hardregion")
222             sys.stderr.write(" %s\n" %‘dok‘)
223             korp.close()
224             for key in ab.keys():
225                 ab[key].close()
226             dbconn.close()
227             sys.exit(1)
228         for hreg_index in range(len(hardregions)):
229             dok = hardregions[hreg_index]
230             fout.write("%s\n" %dok.quelleninfo())
231             fout.write("nach %s\n" %reg_soft)
232             try:
233                 alignedDok = dok.alignedRegion()
234             except AttributeError:
235                 sys.stderr.write("Error: no aligned hardregion for")
236                 sys.stderr.write(" %s\n" %‘dok‘)
237                 korp.close()
```

```
    for key in ab.keys():
        ab[key].close()
    dbconn.close()
    sys.exit(1)
if   korp.getQuellsprache() == "de" and korp.getZielsprache() == "en":
    l1 = eingabe.getRegionen(dok, reg_soft)
    l2 = eingabe.getRegionen(alignedDok, reg_soft)
elif korp.getQuellsprache() == "en" and korp.getZielsprache() == "de":
    l1 = eingabe.getRegionen(alignedDok, reg_soft)
    l2 = eingabe.getRegionen(dok, reg_soft)
elif not wahl in ["WB"]:    # Liste der sprachabhaengigen Abstandsmasse
    l1 = eingabe.getRegionen(dok, reg_soft)        # Reihenfolge egal
    l2 = eingabe.getRegionen(alignedDok, reg_soft)
else:
    sys.stderr.write("Error: cannot compare %s"%korp.getQuellsprache())
    sys.stderr.write(" and %s" %korp.getZielsprache())
    sys.stderr.write(" with %s\n" %`ab[wahl]`)
    korp.close()
    for key in ab.keys():
        ab[key].close()
    dbconn.close()
    sys.exit(1)
if verbose:
    sys.stderr.write("number of regions: %s\n" %`(len(l1),len(l2))`)
laenge1 = len(l1)
laenge2 = len(l2)
if laenge1 <= 1 or laenge2 <= 1:
    mab = ab["NULL"]
else:
    mab = ab[wahl]
mab2 = mab
if leave_out:
    vec1 = [laenge2 - 1, laenge1 - 1]     # Verbindungslinie ol nach ur
    vec2 = [laenge1 - 1, 1 - laenge2]     # senkrecht zur Verbindungsl.
    laenge = math.sqrt(vec2[0]*vec2[0]+vec2[1]*vec2[1])
    if laenge > 0.01:
        vec1[0] = vec1[0] / laenge        # Vektor normieren
        vec1[1] = vec1[1] / laenge
        vec2[0] = vec2[0] / laenge
        vec2[1] = vec2[1] / laenge
    else:
        vec1[0] = 1 / math.sqrt(2)        # default fuer 1:1
        vec1[1] = 1 / math.sqrt(2)
        vec2[0] = 1 / math.sqrt(2)
        vec2[1] = -1 / math.sqrt(2)
    if verbose:
        sys.stderr.write("othonormal vector %s\n" %vec2)
fout.write("%s\t%s\n" %(laenge2, laenge1))
for y in range(laenge1):
    for x in range(laenge2):
        if leave_out:
            dist_x = x * vec1[0] + y * vec1[1]  # Skalarprodukt
            dist_y = x * vec2[0] + y * vec2[1]  # Skalarprodukt
            dxb = min([dist_x, laenge - dist_x])
            lleave_out_dist = leave_out_dist + \
                              leave_out_dist_gradient * dxb
            if lleave_out_dist > leave_out_dist_max >= 0.0:
                lleave_out_dist = leave_out_dist_max
            if math.fabs(dist_y) > lleave_out_dist:
                mab2 = ab["EINS"]
            else:
```

```
                        mab2 = mab
                if text:
                    fout.write("--- %s Nr. %s zu %s---\n" %(reg_soft, y, x))
                    if l1.has_key(y):
                        fout.write(l1[y].getText()+'\n')
                    if l2.has_key(x):
                        fout.write(l2[x].getText()+'\n')
                wert = mab2.zwischen(l1[y], l2[x])
                fout.write("%s\n" %`wert`)
                if text:
                    fout.write('\n')
        if verbose:
            sys.stderr.write(`mab.statistic()`+'\n')
    korp.close()
    for key in ab.keys():
        ab[key].close()
    dbconn.close()
    if aiout:
        aiout.close()

if __name__ == '__main__':
    main()
```

M.3.23 zeitmessung.py

```
#!/usr/bin/python

# Modul zeitmessung

# zwei alternative Basisklassen, fuer Klassen, die die Verweildauer
# in verschiedenen Funktionen messen wollen

import sys
import time
import types

class ZeitmessungNull:

    def __init__(self):
        self.zeit_in = {}

    def statistic(self):
        return self.zeit_in

    def p_begin(self, name):
        pass

    def p_end(self, name):
        pass

class Zeitmessung(ZeitmessungNull):

    def __init__(self):
        ZeitmessungNull.__init__(self)

    def p_begin(self, name):
        if self.zeit_in.has_key(name):
            bis_jetzt = self.zeit_in[name]
        else:
            bis_jetzt = 0.0
        if type(bis_jetzt) == types.FloatType:
```

```
            neu = (bis_jetzt, 0, time.time())   # ..., Ebene 0, aktuelle Zeit
        else:
            bj, level, begin =  bis_jetzt
            level = level + 1
            neu = (bj, level, begin)
        self.zeit_in[name] = neu

    def p_end(self, name):
        bj, level, begin = self.zeit_in[name]
        if level:
            level = level - 1
            neu = (bj, level, begin)
        else:
            neu = bj + (time.time() - begin)   # Differenz zu bj hinzuzaehlen
        self.zeit_in[name] = neu

```

M.4 dds-Paket

Das dds-Paket bildet eine eingenständig übersetzbare Einheit und befindet sind im CVS Baum im Verzeichnis `align`.

M.4.1 Das Makefile align/makefile

```
# Makefile zum Installieren des Aligners in bin
# KOKS - Studienprojekt
# Joachim Wagner, 04. Maerz 2001, 02. Mai 2001

INSTDIR=../bin/

#JAVAC=/usr/lib/jdk1.3/bin/javac
JAVAC=javac
JAVA=java
JAR=jar
CC=gcc
#CC=cc
#CFLAGS=-Wall -O2 -pedantic -m486 -malign-loops=2 -malign-jumps=2 -malign-functions=2 -DCPU=586
CFLAGS=-Wall -pedantic -m486 -malign-loops=2 -malign-jumps=2 -malign-functions=2 -DCPU=586


dummy:
   @echo "make all      kompiliert die Programme (cgaligner, ddsaligner, ddskommandozeile) lokal"
   @echo "make install  kopiert Programm nach $(INSTDIR)"
   @echo "make clean    räumt (derzeit nur dds) auf"
   @echo "make ddshelp  gibt Auskunft über die dds-Targets"

all:  align dds

install: all
   cp align $(INSTDIR)

clean:   ddsclean
   rm -f align align.o

align:   align.c
   $(CC) $(CFLAGS) -o $@ -lm align.c

ddshelp:
   @echo "dds kompiliert (grafisches) Mavis und Kommandozeilenversion"
   @echo "ddsdemomatrix erzeugt Zufallsmatrix in der Datei zufall.mtr"
```

```
   @echo "ddsclean entfernt .class und sonstige Dateien"
   @echo "ddsjar erzeugt je eine .jar Datei für Mavis (mavis.jar) und Kommandozeilenaligner (al.jar)"
   @echo "al.jar ist nicht komprimiert, da dies die Ladegeschwindigkeit erhöht "
   @echo "ddsrunmavis zeigt, wie Mavis aus der jar-Datei zu starten ist"
   @echo "ddsrunal demonstriert das gleiche für die Kommandozeilenversion"


dds:  ddsbeobachtung ddsmatrix ddsmatrixAligner ddsmavis ddskommandozeile ddsjar

ddsmavis:   de/denkselbst/mavis/StutzenDialog.class de/denkselbst/mavis/Mavis.class de/denkselbst/mavis/MACV.

ddsmatrixAligner: de/denkselbst/matrixAligner/AlignmentObserver.class de/denkselbst/matrixAligner/BFSMatrixAl

ddsbeobachtung:   de/denkselbst/beobachtung/Beobachtbar.class de/denkselbst/beobachtung/Beobachter.class

ddsmatrix:  de/denkselbst/matrix/Matrix.class de/denkselbst/matrix/MatrixView.class de/denkselbst/matrix/Writ

ddskommandozeile: de/denkselbst/kommandozeile/Align.class de/denkselbst/kommandozeile/HeadingMapper4Arno.clas

%.class: %.java
   $(JAVAC) -classpath lib/adt-20010208.jar:. $*.java

ddsjar:
   #$(JAR) cmvf de/denkselbst/mavis/manifest mavis.jar lib/adt-20010208.jar de/denkselbst/beobachtung/*.class
   #$(JAR) cmvf0 de/denkselbst/kommandozeile/manifest al.jar -C lib/adt-20010208.jar de/denkselbst/beobachtun
   $(JAR) cmf de/denkselbst/mavis/manifest mavis.jar com/bluemarsh/adt/ de/denkselbst/beobachtung/ de/denksel
   $(JAR) cmf0 de/denkselbst/kommandozeile/manifest al.jar com/bluemarsh/adt/ de/denkselbst/beobachtung/*.cla
   $(JAR) cmf0 de/denkselbst/kommandozeile/manifest2 headingMapper.jar de/denkselbst/beobachtung/*.class de/d

ddsclean:
   -rm -f de/denkselbst/mavis/*.class
   -rm -f de/denkselbst/matrix/*.class
   -rm -f de/denkselbst/matrixAligner/*.class
   -rm -f de/denkselbst/beobachtung/*.class
   -rm -f de/denkselbst/kommandozeile/*.class
   -rm -f zufall.mtr
   -rm -f al.jar
   -rm -f mavis.jar
   -rm -f headingMapper.jar

ddsdemomatrix:
   @echo Schreibe 17x14 Zufallsmatrix: zufall.mtr
   $(JAVA) -classpath lib/adt-20010208.jar:. de.denkselbst.matrix.WriteRandomMatrix 17 14 > zufall.mtr

ddsrunmavis:
   $(JAVA) -jar mavis.jar

ddsrunal:
   $(JAVA) -jar al.jar
```

M.5 Java

M.5.1 de.denkselbst.matrix.Matrix.java

```
package de.denkselbst.matrix;

import de.denkselbst.beobachtung.Beobachtbar;
import de.denkselbst.beobachtung.Beobachter;

import java.util.ArrayList;
```

```
import java.util.Arrays;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.FileReader;



/**
Diese Klasse repräsentiert eine aus double Werten bestehende Abstandsmatrix.
Die Abstandsmatrix als solche ist die Schnittstelle zwischen Abstandsmaß und
Aligner.
Daher ist diese Klasse ein sehr wichtiger und zentraler Bestandteil des
Aligners. Neben den Abstandswerten enthalten Objekte dieser Klasse zwei
Strings, die den Pfad zur Ausgangsdatei und Zusatzinformationen enthalten
(quelle, info).

Im Allgemeinen werden Objekte dieser Klasse über einen Konstruktor mit
Dateinamen aus einer Datei, bzw. aus STDIN mit Werten gefüllt. Das benutzte
Dateiformat hat folgende Form:
Quelle-String \n
Info-String \n
Breite Hoehe \n
Wert1 \n Wert2 \n ... WertN

Diese Klasse erkennt neben \n auch anderen Whitespace als Trenner zwischen den
Abstandswerten an. (Siehe entsprechende Methode.)

Die Abstandswerte, so wurde es im Projekt vereinbart, liegen im Intervall von
0 bis 1. Damit die Heuristik des A*-Aligners ihre Funktionalität erfüllen
kann, müssen alle Abstandswerte > 0 sein. Die Methode
normalisiereDich() verschiebt die Abstandswerte einer Matrix in das Intervall
[0.1 ... 1]. Dies erhöht auch den Kontrast in einer Matrix aus TG-Werten, die
in einem sehr kleinen Intervall liegen.

Eine Reihe von Methoden mit zugegebenermaßen recht ausschweifigen Namen
erlauben den Zugriff auf die Verkapselten Daten.
Am wichtigsten sind holeDatumVonStelle(int x, int y) und
wasSindVonHierDieUnterschaetztenKostenZurRechtenUnterenEcke().

In dem GUI-Tool Mavis gibt es viele Objekte, die sich für Änderungen
(Transformation der Daten, neue Datei geladen) interessieren, daher
wurde ein einfaches Observer-Modell geschaffen (package
de.denkselbst.beobachtung), das vom Matrix-Objekt unterstüzt wird.

Sonstiges:
Die Zugriffsmethoden sind derzeit nicht synchronisiert.


*/
public class Matrix implements Beobachtbar{

   /** beinhaltet aus Datei gelesene Quellenangabe */
   protected String quelle;

   /** beinhaltet aus Datei gelesene Info-Zeile */
   protected String info;

   /** merkt sich den Dateinamen, dieser wird hier von der
       Methode ReadFromFile hinterlegt, bzw. von dateilosen
```

```
    Konstruktoren auf "" gesetzt*/
protected String filename;

/** beinhaltet die Daten der Matrix */
protected double[][] daten;

/** merkt sich den kleinsten in der Matrix eingetragenen Wert */
protected double minimum;
/** Koordinaten des kleinsten Werts */
protected int minXPos, minYPos;

/** merkt sich den größten in der Matrix eingetragenen Wert */
protected double maximum;
/** Koordinaten des größten Werts */
protected int maxXPos, maxYPos;


/** gibt es schon einen kleinsten Wert? */
protected boolean minimumBereitsErmittelt;

/** Maximum schon ermittelt? */
protected boolean maximumBereitsErmittelt;


/**
  * Parameterloser Konstruktor:
  * sollte nur verwendet werden, wenn Matrix aus stdin gefüllt wird!
  */
public Matrix(){
   daten = new double[1][1];
   minimumBereitsErmittelt = false;
   filename="uninitialisierte Matrix!!!";
}

/** legt ein Matrix-Objekt mit einer leeren Matrix an
       * @param xDim die gewünschte Breite
       * @param yDim die gewünschte Höhe
  */
public Matrix(int xDim, int yDim){
   daten = new double[xDim][yDim];
   minimumBereitsErmittelt = false;
   filename="";
}

/**
  * legt ein Matrix-Objekt an
  * @param m die Daten
  */
public Matrix(double[][] m){
   daten = m;
   minimumBereitsErmittelt = false;
   filename="";
}

/**
  * legt ein Matrix-Objekt aus einer Datei an
  * @param datei der Dateiname
  */
public Matrix(String file){
   daten = readFromFile(file);
   minimumBereitsErmittelt = false;
```

```
}



/**
  * liest Daten aus einer Datei ein und liefert double[][]
  * (bemüht readFromBufferedReader())
  * @param file Dateiname als String
  */
private double[][] readFromFile(String file){
   filename = file;  // Dateinamen merken

   try{  // BufferedReader anlegen und an zuständige Read-Methode übergeben,
      // deren Ergebnis liefern
      BufferedReader in = new BufferedReader(new FileReader(file));
      return readFromBufferedReader(in);
   }catch(IOException e){  // IO-Fehler
      System.err.println(e.getMessage());
      System.err.println("readFromFile: Fehler beim Lesen der Datei "+file+
      " . Liefere leere 6x6 Matrix.");
      return new double[6][6]; // macht readfromBufferedreader doch eh?!
   }


}

/**
  * liest Daten von StdIn ein und liefert double[][]
  * (bemüht readFromBufferedReader())
  */
private double[][] readFromStdIn(){
   filename="von_stdin_gelesen";
   BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
   return readFromBufferedReader(in);
}

/**
  * liest Daten von einem BufferedReader ein und liefert double[][]
  * (bemüht readFromBufferedReader())
  */
private double[][] readFromBufferedReaderJoachim(BufferedReader bf){
   filename="von_BufferedReader_gelesen";
   return readFromBufferedReader(bf);
}



/**
  * liest Daten aus BufferedReader und liefert double[][]
  * (macht die eigentliche Arbeit für die beiden anderen Read-Methoden)
  * @param in der BufferedReader
  */
  private double[][] readFromBufferedReader(BufferedReader in){
   int x = 0;                              // Abmessungen der Matrix
   int y = 0;                              //  "
   double[][] matrix = new double[0][0];  // Referenz auf Matrix

   try{
      quelle = in.readLine();
      info = in.readLine();
      String line = in.readLine();
      StringTokenizer tok = new StringTokenizer(line, "\t \n", false);
```

```
      // nächstes Token besorgen
      if(!tok.hasMoreTokens()){
         while(!tok.hasMoreTokens() && line!=null){
            line = in.readLine();
            tok = new StringTokenizer(line, "\t \n", false);
         }
      }
      // breite der Matrix holen
      x = Integer.parseInt(tok.nextToken());

      // nächstes Token besorgen
      if(!tok.hasMoreTokens()){
         while(!tok.hasMoreTokens() && line!=null){
            line = in.readLine();
            tok = new StringTokenizer(line, "\t \n", false);
         }
      }
      // Höhe der Matrix holen
      y = Integer.parseInt(tok.nextToken());

      // Matrix anlegen
      matrix = new double[x][y];
      String token = null;

      // Zeilenweise einlesen
      for (int j = 0; j < y; j++){
         for (int i = 0 ; i < x ; i++){
            // Nächstes Token besorgen
            if(!tok.hasMoreTokens()){
               while(!tok.hasMoreTokens() && line!=null){
                  line = in.readLine();
                  tok = new StringTokenizer(line, "\t \n", false);
               }
            }
            token = tok.nextToken();
            matrix[i][j] = Double.parseDouble(token);
            //System.err.println("habe "+(i+(j*x))+" von "+(x*y)+
            //" Zahlen eingelesen.");
         }
      }
   }catch(IOException e){
      System.err.println(e.getMessage());
      System.err.println(" ReadFromBufferedReader: IO-Fehler. Liefere leere"+
               " 6x6 Matrix.");
      return new double[6][6];
   }

   return matrix; // Die Matrix liefern
}



/**
  * liefert den Quellen-String
  */
public String holeQuelle(){
   return quelle;
}
```

```
/**
  * liefert den Info-String
  */
public String holeInfo(){
   return info;
}


/**
  * liefert den Dateinamen als String
  */
public String holeDateiname(){
   return filename;
}



/**
  * liefert die Breite der Matrix
  */
public int holeBreite(){
   return daten.length;
}

/**
  * liefert die Hoehe der Matrix
  */
public int holeHoehe(){
   return daten[0].length;
}


/**
  * Kopie von daten[][] liefern. Eine solche Kopie der daten[][] kann von
  * anderen Objekten ohne Auswirkungen auf andere Programmteile destruktiv verändert
  * werden.
  */
public double[][] kopieDerDaten(){
   double[][] kopie = new double[holeBreite()][holeHoehe()];

   for(int i = 0; i < holeBreite(); i++){
      for(int j = 0; j < holeHoehe(); j++){
         kopie[i][j] = daten[i][j];
      }
   }

   return kopie;
}

/**
  * liefert den Wert der Matrix an der Stelle (x,y)
  * @param x die x-Position
  * @param y die y-Position
  */
public double holeDatumVonStelle(int x, int y){
   return daten[x][y];
}

/**
  * setzt den Wert der Matrix an der Stelle (x,y) auf datum
  * Minimum und Maximum werden gepflegt
```

```
    * @param x die x-Position
    * @param y die y-Position
    * @param datum der zu setzende double Wert
    */
   public void setzeDatumAnStelle(int x, int y, double datum){
      // datum eintragen
      daten[x][y] = datum;
      // handelt es sich um ein neues Minumum?
      if (minimumBereitsErmittelt && datum < minimum) minimum = datum;
      if (maximumBereitsErmittelt && datum > maximum) maximum = datum;
      // Beobachtern Meldung von der Änderung der Matrix machen...
      beobachterBenachrichtigen("Habe "+datum+
                                " an Stelle ("+x+", "+y+") gesetzt.");
   }


   /**
    * intern aufrufen, um Zustandsänderung klarzumachen
    * altes Minimum und Maximum verlieren Gültigkeit
    * z.B. nach Transformation und Stutzen!
    */
   protected void geaendert(){
      minimumBereitsErmittelt = false;
      maximumBereitsErmittelt = false;
      //minimum = -999; // Weg damit!
      //maximum = +999; // "
      //minXPos = -666;
      //minYPos = -666;
   }

   /**
    * Kleinsten Wert (Minimum) der Matrix liefern
    *
    */
   public double wasIstDerKleinsteWert(){
      if(!minimumBereitsErmittelt){
         minimum = besorgeMinimum();
         minimumBereitsErmittelt = true;
      }
      return minimum;
   }

   /** x-Koordinate des Minimums liefern */
   public int getMinX(){
      besorgeMinimum(); // Falls Wert nicht im Cache, Besorgung veranlassen.
      return minXPos;
   }

   /** y-Koordinate des Minimums liefern */
   public int getMinY(){
      besorgeMinimum(); // Falls Wert nicht im Cache, Besorgung veranlassen.
      return minYPos;
   }

   /**
    * Matrix zellenweise durchgehen und dabei den
    * kleinsten Wert (Minimum) bestimmen
    */
   private double besorgeMinimum(){
      minXPos = 0;
      minYPos = 0;
```

```
   double min = daten[0][0];
   for(int i = 0; i < holeBreite(); i++){
      for(int j = 0; j < holeHoehe(); j++){
         if(daten[i][j] < min){
            min = daten[i][j];
            minXPos = i;         // Koordinaten merken
            minYPos = j;         //  "
         }
      }
   }
   return min;
}


/**
  * Maximum der Matrix liefern
  *
  */
public double wasIstDerGroessteWert(){
   if(!maximumBereitsErmittelt){
      maximum = besorgeMaximum();
      maximumBereitsErmittelt = true;
   }
   return maximum;
}


/**
  * Matrix zellenweise durchgehen und dabei den
  * größten Wert (Maximum) aufsammeln
  */
private double besorgeMaximum(){
   double max = daten[0][0];
   for(int i = 0; i < holeBreite(); i++){
      for(int j = 0; j < holeHoehe(); j++){
         if(daten[i][j] > max){
            max = daten[i][j];
            maxXPos = i;         // Koordinaten merken
            maxYPos = j;         //  "
         }
      }
   }
   return max;
}



/**
  * Unterschätzte Kosten zur rechten unteren Ecke liefern
  * (Diagonale Distanz)
  */
public double wasSindVonHierDieUnterschaetztenKostenZurRechtenUnterenEcke(
                         int xPos, int yPos){
   return Math.max( Math.abs((holeBreite()-1) - xPos),
          Math.abs((holeHoehe()-1) - yPos) )
                    *  wasIstDerKleinsteWert();
}

/**
  * Mindestkosten von x1,y1 nach x2,y2 liefern
  */
```

```
public double minKost(int x1, int y1, int x2, int y2){
   return Math.max( Math.abs((x2-x1)),
          Math.abs(y2-y1)) *
          wasIstDerKleinsteWert();
}

/**
  * minimale Anzahl Schritte von xPos, yPos zum Ziel
  * (rechte untere Ecke)
  */
public int wievieleSchritteVonHierZurRechtenUnterenEcke(int xPos, int yPos){
   return Math.max( Math.abs((holeBreite()-1) - xPos),
          Math.abs((holeHoehe()-1) - yPos) );
}

/**
  *   Matrix auf die gegebenen Dimensionen zurechtstutzen
  **/
public void stutzenAuf(int x, int y){
   // bei Gelegenheit Sicherheitsüberprüfungen einbauen!

   double[][] neu = new double[x][y];
   for(int i = 0; i < x; i++){
      for(int j = 0; j < y; j++){
         neu[i][j]=daten[i][j];
      }
   }

   daten = neu;
   geaendert();   // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
   // die Beobachter benachrichtigen
   beobachterBenachrichtigen("Habe die Matrix auf "+x+" * "+y+" gestutzt.");
}

/**
  * Eine komplette Spalte der Matrix auf Wert w setzen
  */
public void fillColumn(int x, double w){
   if(0 <= x && x < holeBreite()){
      for(int j = 0; j < holeHoehe(); j++){
         daten[x][j] = w;
      }
   geaendert();   // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
   // die Beobachter benachrichtigen
   beobachterBenachrichtigen("Habe alle Zellen der Spalte"+x+" auf "+w+
              " gesetzt.");
   }
}

/**
  * Eine komplette Zeile der Matrix auf Wert w setzen
  */
public void fillRow(int y, double w){
   if(0 <= y && y < holeHoehe()){
      for(int i = 0; i < holeBreite(); i++){
         daten[i][y] = w;
      }
   geaendert();   // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
   // die Beobachter benachrichtigen
   beobachterBenachrichtigen("Habe alle Zellen der Zeile"+y+" auf "+w+
              " gesetzt.");
```

```
   }
}



/**
  * Mittelwert berechnen
  */
public double berechneMittelwert(){
   double sum = 0;
   for(int i = 0; i < holeBreite(); i++){
      for(int j = 0; j < holeHoehe(); j++){
         sum+=daten[i][j];
      }
   }
   return sum / (holeBreite() * holeHoehe());
}



/**
  * Varianz berechnen
  */
public double berechneVarianz(){
   double mittelwert = berechneMittelwert();

   double sum = 0;
   for(int i = 0; i < holeBreite(); i++){
      for(int j = 0; j < holeHoehe(); j++){
         sum+= ((daten[i][j] - mittelwert) * (daten[i][j] - mittelwert));
      }
   }
   return sum / ((holeBreite() * holeHoehe())-1);
}

/**
  * Standardabweichung berechnen
  */
public double berechneStandardabweichung(){
   return Math.sqrt(berechneVarianz());
}



/**
  * Transformation: Daten normalisieren auf Intervall [0.1 .. 1]
  */
public void normalisiereDich(){
   int breite = holeBreite();
   int hoehe = holeHoehe();
   double min = wasIstDerKleinsteWert();
   double max = wasIstDerGroessteWert();

   double transformiert[][] = new double[breite][hoehe];

   for(int i = 0; i < holeBreite(); i++){
      for(int j = 0; j < holeHoehe(); j++){
         // in Bereich [0 .. 1] skalieren
         if(max-min != 0){
            transformiert[i][j] = (daten[i][j] - min) * (1.0 / (max - min));
         }else transformiert[i][j]=42;
```

```
            // in Bereich [0.1 .. 1] verschieben
            transformiert[i][j] = transformiert[i][j] * (0.9) + 0.1;
         }
      }
      daten = transformiert;
      geaendert();   // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
      beobachterBenachrichtigen("Matrixwerte auf Intervall [0.1 .. 1] skaliert");
   }


   /**
     * Daten transformieren: Clusterbildung nach Rangfolge der Daten
     */
   public void rangfolgenClustering(int anzahlKlassen){

      int breite = holeBreite();
      int hoehe = holeHoehe();
      int anzahlWerte = breite * hoehe;

      // jede Klasse bekommt als Wert ein Vielfaches von diesem Wert
      double werteSchrittweite = 1.0 / (anzahlKlassen + 1); // +1 für eine
            // Dummy-Klasse, damit kein Wert mehr auf 0.0 abgebildet wird...

      double[] werte = new double[anzahlWerte];
      double[][] neu = new double[breite][hoehe];

      for(int i = 0; i < breite; i++){
         for(int j = 0; j < hoehe; j++){
          werte[(j*breite)+i] = daten[i][j];
         }
      }

      // So, Werte sind sortiert!
      Arrays.sort(werte);
      // Alle Werte durchgehen
      int klasse = 1;
      int kCount = 0;
      int werteProKlasse = anzahlWerte / anzahlKlassen;
      for(int i = 0; i < werte.length; i++){
         kCount++;
         if(kCount > werteProKlasse){ klasse++; kCount=0;}

         repl(werte[i],klasse * werteSchrittweite, neu);
      }
      daten = neu;
      geaendert();   // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
      beobachterBenachrichtigen("bogoRangfolgenClustering durchgeführt");
   }

   /** schlampige replace-Funktion; ersetzt alle Vorkommen von w1
     * in der Matrix durch w2 im Argument
          */
   private void repl(double w1, double w2, double[][] x){
      for(int i = 0; i < holeBreite(); i++){
         for(int j = 0; j < holeHoehe(); j++){
            if(daten[i][j]==w1) x[i][j]=w2;
         }
      }
   }
```

```
/**
  * Matrix mit neuen Daten aus Datei füttern
  * @param file String mit Dateiname
  */
public void setzeDatenAusDatei(String file){
   // Daten besorgen
   daten = readFromFile(file);
   // Kleinster Wert dieser neuen Daten ist logischerweise noch nicht ermittelt
   // worden
   minimumBereitsErmittelt = false;
   // Beobachtern Meldung davon machen...
   beobachterBenachrichtigen("Habe potentiell neue Daten aus Datei "+file+
              " eingelesen.");
}

/**
  * Matrix mit neuen Daten aus Stdin füttern
  */
public void setzeDatenAusStdin(){
             BufferedReader bf = new BufferedReader(new
         InputStreamReader(System.in));
   setzeDatenAusGepuffertemLeser(bf);
}

/**
  * Matrix mit neuen Daten aus BufferedReader füttern
  */
public void setzeDatenAusGepuffertemLeser(BufferedReader br){
   // Daten besorgen
   daten = readFromBufferedReaderJoachim(br);
   // Kleinster Wert dieser neuen Daten ist logischerweise noch nicht ermittelt
   // worden
   minimumBereitsErmittelt = false;
   // Beobachtern Meldung davon machen...
   beobachterBenachrichtigen("Habe potentiell neue Daten von stdin gelesen.");
}



/**
  * Matrix als String liefern (billige Ausgabe zu Diagnosezwecken)
  */
public String toString(){
   int b = holeBreite();
   int h = holeHoehe();
   StringBuffer buf = new StringBuffer("Matrix \n   Breite: "+b+
         "   Höhe: "+h+"\n   Daten\n");

   for(int j = 0; j < h; j++){
      buf.append("      ");
      for(int i = 0; i < b; i++){
         buf.append( holeDatumVonStelle(i,j) +" \t");
      }
      buf.append("\n");
   }

   return buf.toString();
}

/**
```

```
     * nimmt die Objekte auf, die Interesse an
     * Änderungen der Matrix angemeldet haben
     */
   private ArrayList dieBeobachter;

   /**
     * Einen neuen Beobachter in die Liste der im Falle einer
     * Änderung zu benachrichtigen Objekte aufnehmen.
     * @param b das Objekt, das Interesse an den Änderungen der Matrix hat
     */
   public void beobachterAnmelden(Beobachter b){
      if(dieBeobachter == null) dieBeobachter = new ArrayList();
      dieBeobachter.add(b);
   }

   /**
     * einen Beobachter aus der Liste der zu benachrichtigen
     * Objekte entfernen.
     * @param b das Objekt, das das Interesse an den
     * Änderungen der Matrix verloren hat
     */
   public void beobachterAbmelden (Beobachter b){
      if(dieBeobachter != null) dieBeobachter.remove(b);
   }

   /**
     * alle Beobachter ob einer Änderung benachrichtigen, Referenz
     * auf das Beobachtete Objekt (this)
     * und Nachricht String werden mitgeliefert
     * @param nachricht String, der zur Beschreibung der Änderung
     * benutzt werden kann
     */
   public void beobachterBenachrichtigen(String nachricht){

      if(dieBeobachter == null) return;   // keine Beobachter, keine Nachricht.

      for(int i=0; i<dieBeobachter.size(); i++){
         if(dieBeobachter.get(i) != null){   // Wenn da überhaupt jemand ist (sollte
                        //immer so sein!)
            ((Beobachter)dieBeobachter.get(i)).beobachtetesObjektHatSichGeregt(this,
nachricht);
         }
         else{ // NULL-Referenzen löschen! sollte alles nie passieren!!!
            System.out.println("Matrix: NULL-Beobachter entfernt!");
            dieBeobachter.remove(i);
         }
      }
   }

   /** winziger Test: Lese Matrix aus Datei args[0] ein
     * und gebe sie auf System.out aus.
          */
   public static void main (String[] args){
      Matrix m = new Matrix(args[0]);
      System.out.println(m);
   }

}
```

M.5.2 de.denkselbst.matrix.Matrix.Pfad.java

```
package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.Iterator;

/**
Diese Klasse repräsentiert eine Liste von Koordinatenpaaren.

Ein Pfad ist eine Liste von Koordinatenpaaren. Im Rahmen von
Graphensuchalgorithmen ist das ein geeignetes Mittel, um einen Pfad in einer
Matrix zu beschreiben. Längere horizontale Abschnitte in einem Pfad werden als
n:m - Alignments aufgefasst, durch eine Konvertierung des Pfades in eine
KoksZuordnung wird dem Rechnung getragen.
MatrixAligner liefern ihre Ergebnisse derzeit als Pfade.

Pfade wissen, wie sie sich in KoksZuordnungen konvertieren können.

Die Klasse KoksZuordnung bietet eine Konstruktor, der einen Pfad
entgegen nimmt. Die KoksZuordnung ruft dann die Methode toWagnerString beim
Pfad auf und erhält so einen String, den sie interpretieren kann. Quelle und
Info müssen aus einer Matrix übernommen werden, deshalb hat die
toWagnerString Methode Argumente, in denen diese Strings übergeben werden
müssen.



*/

public class Pfad{

   /**
     * Kapselt x- und y-Koordinate
     */
   public static class Koordinate{
      /** enthält x-Koordinate*/
      public int x;
      /** enthält y-Koordinate*/
      public int y;

      /**
        * legt eine Koordinate an
        * @param x - x-Koordinate
        * @param y - y-Koordinate
        */
      public Koordinate(int x, int y){
         this.x = x;
         this.y = y;
      }
   }

   /** beinhaltet die Koordinaten, aus denen der Pfad besteht */
   ArrayList p;

   /** neues Pfad Objekt anlegen */
   public Pfad(){
      p = new ArrayList();
   }

   /** eine Koordinate ans Ende des Pfades anhängen
```

```
  * @param x - x-Koordinate
  * @param y - y-Koordinate
  */
public void haengeAn(int x, int y){
   p.add(new Koordinate(x, y));
}

/** eine Koordinate an den Anfang des Pfades anhängen
  * @param x - x-Koordinate
  * @param y - y-Koordinate
  */
public void haengeVorneAn(int x, int y){
   p.add(0, new Koordinate(x, y));
}

/** einen Iterator über die Elemente des Pfades liefern*/
public Iterator iterator(){
   return p.iterator();
}

/** Pfade umdrehen*/
public void umdrehen(){
   if(p.size() == 0) return;
   ArrayList p2 = new ArrayList(p.size());
   for(int i = 0; i < p.size(); i++){
      p2.add( p.get(p.size()-1-i));
   }
   p = p2;
}

/**
  * Einfache Ausgabe
  */
public String toString(){
   StringBuffer buf = new StringBuffer();
   Koordinate k;
   for(int i = 0; i < p.size(); i++){
      k = (Koordinate)p.get(i);
      buf.append("["+k.x+","+k.y+"] ");
   }
   return buf.toString();
}

/**
  * Pfad als String im Wagner-Format liefern.
  * Verwendungszweck ist sowohl textuelle Ausgabe, als auch
  * Schritt bei der Konvertierung in KoksZurodnung.
  * Die Strings quelle und info müssen übergeben werden, Matrix-
  * Objekte verfügen über entsprechende Strings, von dort sollten
  * diese Informationen also beschafft werden.
  *
  * Der Algorithmus verfolgt den Pfad von Anfang bis Ende und verwandelt
  * ihn in eine Folge von Segment-Zuordnungen.
  * Eine Diagonalbewegung nach rechts unten schliesst die aktuelle
  * Segmentzuordnung ab und beginnt eine neue. Durch vertikalen oder
  * horizontalen Pfadverlauf verknüpfte Sätze werden an die aktuelle
  * Segment-Zuordnung angehängt.
  */
public String toWagnerString(String quelle, String info){
   if(p.size()==0) return ""; // leerer Pfad = leerer Wagner-String
```

```
StringBuffer buf = new StringBuffer(quelle+"\n"+info+"\n");

int segment = 0;
Koordinate k;

int RECHTS = 1;         // einige Konstanten
int RUNTER = 2;
int DIAGONAL = 3;
int UNBEKANNT = -9;

// Vorbereitungen
ArrayList seg1 = new ArrayList(2);  // Satznummern Sprache 1
ArrayList seg2 = new ArrayList(2);  // Satznummern Sprache 2

k = (Koordinate)p.get(0);
int altX = k.x;
int altY = k.y;

seg2.add(new Integer(k.x));
seg1.add(new Integer(k.y));

int altRichtung = UNBEKANNT;
int neueRichtung = UNBEKANNT;

// alle Koordinaten abwandern
for(int i = 1; i < p.size(); i++){

   k = (Koordinate)p.get(i);

   int neuesX = k.x;
   int neuesY = k.y;

   // Laufrichtung ermitteln
   if( (neuesX == altX) && (neuesY != altY) ){  // Der Pfad führt nach unten
      neueRichtung = RUNTER;
   }
   else if( (neuesX != altX) && (neuesY == altY) ){// nach rechts
      neueRichtung = RECHTS;
   }
   else{                                                  // Diagonal
      neueRichtung = DIAGONAL;
   }

   // Wenn’s diagonal geht, dann sofort neue Ausgabe!
   if(neueRichtung == DIAGONAL){
      // Ausgabe, Listen zurücksetzen
         // Anfang auseben
         buf.append(""+segment+"\t[");
         for(int t=0; t<seg1.size(); t++){
            buf.append(""+seg1.get(t)+",");
         }
         // Überschüssiges Komma entfernen
         buf.deleteCharAt(buf.length()-1);
         // Nächste Spalte
         buf.append("]\t[");
         for(int t=0; t<seg2.size(); t++){
            buf.append(""+seg2.get(t)+",");
         }
         // Überschüssiges Komma entfernen
         buf.deleteCharAt(buf.length()-1);
         // Klammer zu und \newline
```

```
            buf.append("]\n");

            // Nächste Runde nächstes Segment verarzten
            segment++;

            seg1 = new ArrayList(2);   // Satznummern Sprache 1
            seg2 = new ArrayList(2);   // Satznummern Sprache 2


         altRichtung = UNBEKANNT;

         seg2.add(new Integer(k.x));
         seg1.add(new Integer(k.y));

      }
      else if(altRichtung == UNBEKANNT){  // alte Richtung unbekannt
         if(neueRichtung == RUNTER){
            seg1.add(new Integer(k.y));
         }

         if(neueRichtung == RECHTS){
            seg2.add(new Integer(k.x));
         }

         altRichtung = neueRichtung;
      }
      else{ // Alte Richtung bekannt

         if( neueRichtung == RECHTS){
            seg2.add(new Integer(k.x));
         }
         else if(neueRichtung == RUNTER){
            seg1.add(new Integer(k.y));
         }
      }


      altX = k.x; // Koordinaten für die nächste Iteration aufheben
      altY = k.y; //  "
   }// Schleifenende


   // Ausgeben, was noch in den Puffern steht.
            // Anfang ausgeben
            buf.append(""+segment+"\t[");
            for(int t=0; t<seg1.size(); t++){
               buf.append(""+seg1.get(t)+",");
            }
            // Überschüssiges Komma entfernen
            buf.deleteCharAt(buf.length()-1);
            // Nächste Spalte
            buf.append("]\t[");
            for(int t=0; t<seg2.size(); t++){
               buf.append(""+seg2.get(t)+",");
            }
            // Überschüssiges Komma entfernen
            buf.deleteCharAt(buf.length()-1);
            // Klammer zu und \newline
            buf.append("]\n");
```

```
   return buf.toString();  // Ergebnis liefern
}



/**
  * Demonstration / Test
  */
public static void main(String[] args){
   System.out.println("Test 1");
   Pfad p = new Pfad();
   p.haengeAn(0,0);
   p.haengeAn(0,1);
   p.haengeAn(1,2);
   p.haengeAn(2,3);
   p.haengeAn(2,4);
   p.haengeAn(3,5);
   p.haengeAn(3,6);

   System.out.println("toString:");
   System.out.println(p.toString());

   System.out.println("");
   System.out.println("toWagner:");
   System.out.println(p.toWagnerString("generiert","Test"));


   p = new Pfad();
   p.haengeAn(0,0);
   p.haengeAn(1,0);
   p.haengeAn(2,1);
   p.haengeAn(3,1);
   p.haengeAn(4,1);
   p.haengeAn(5,2);
   p.haengeAn(6,2);

   System.out.println("");
   System.out.println("Test 2");
   System.out.println("toString:");
   System.out.println(p.toString());

   System.out.println("");
   System.out.println("toWagner:");
   System.out.println(p.toWagnerString("generiert","Test"));


   p = new Pfad();
   p.haengeAn(0,0);
   p.haengeAn(1,1);
   p.haengeAn(2,2);
   p.haengeAn(3,3);
   p.haengeAn(4,4);
   p.haengeAn(5,5);
   p.haengeAn(6,6);

   System.out.println("");
   System.out.println("Test 3");
   System.out.println("toString:");
   System.out.println(p.toString());

   System.out.println("");
```

```
      System.out.println("toWagner:");
      System.out.println(p.toWagnerString("generiert","Test"));
   }

}
```

M.5.3 de.denkselbst.matrix.Matrix.KoksZuordnung.java

```
package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.StringReader;

/**
Diese Klasse repräsentiert ein Alignment. Ein Alignment besteht aus
Zuordnungen (KoksZuordnung.Zuordnung), die n Sätze (Segmente) aus L1 mit m Sätzen aus L2
verbinden. Eine Liste solcher Zuordnungen, die sequentiell die Menge aller
Sätze bedeckt, ist eine KoksZuordnung.
KoksZuordnungen können im Wagner-Format textuell ausgegeben werden und
stellen damit die Schnittstelle vom Aligner zu den nachfolgenden
Modulen dar.
*/

public class KoksZuordnung {

   /* Innere Klasse verkapselt die Zuordnung
    * von zwei Satzmengen.
    */
   public static class Zuordnung{

      private ArrayList lang1, lang2;

      /** Konstruktor */
      public Zuordnung(){
         lang1 = new ArrayList(2);
         lang2 = new ArrayList(2);
      }

      /** Füge Liste 1 eine (Satz)Nummer hinzu*/
      public void addL1(int i){
         lang1.add(new Integer(i));
      }

      /** Füge Liste 2 eine (Satz)Nummer hinzu*/
      public void addL2(int i){
         lang2.add(new Integer(i));
      }

      /** Liefere Liste1 nach aussen (z.B. wird damit an anderer Stelle
          eine HTML-Tabelle mit den tatsächlichen  Sätzen erzeugt) */
      public ArrayList getL1(){
         return lang1;
      }

      /** Liefere Liste2 nach aussen (z.B. wird damit an anderer Stelle
```

```
        eine HTML-Tabelle mit den tatsächlichen  Sätzen erzeugt) */
   public ArrayList getL2(){
      return lang2;
   }

   /** Gibt die Zuordnung im Wagnerformat aus*/
   public String toString(){
      // Auch hier wird erst die Y-Seite, dann die X-Seite ausgegeben,
      // damit das Ergebnis wieder stimmt....
      // Nicht geprüft, ob die Vertauschung hier wirklich notwendig,
      // wenn es also wieder Probleme gibt, erst hier nachsehen!

      StringBuffer buf = new StringBuffer("[");

      for(int i = 0; i < lang2.size(); i++){
         buf.append( lang2.get(i) );
         buf.append(",");
      }
      buf.deleteCharAt(buf.length()-1);

      buf.append("]\t[");

      for(int i = 0; i < lang1.size(); i++){
         buf.append(  lang1.get(i) );
         buf.append(",");
      }
      buf.deleteCharAt(buf.length()-1);

      buf.append("]");

      return buf.toString();
   }

   /**
     * Trägt in der Liste koord die entsprechenden Koordinaten der Zuordnung ein
     * x1,y1,x2,y2,...xn,yn
     */
   public void betroffeneKoordinaten(ArrayList koord){

      for(int i = 0; i < lang1.size(); i++){
         for(int j = 0; j < lang2.size(); j++){
         // Y ist X und X ist Y, habe ich gerade bemerkt...
         // Also erst kommen in der eingelesenen Datei Y-Werte, dann X-Werte
         // Da ich die falschrum einlese, gebe ich sie hier auch falschrum wieder
         // aus, und dann ist ALLES richtig... :-)
            koord.add(lang2.get(j));
            koord.add(lang1.get(i));
         }
      }
   }


}

private String quelle;
private String info;

private ArrayList zuordnungen;

/**
  * Aus Datei Kokszuordnung einlesen
```

```
  */
public KoksZuordnung(String file){
   zuordnungen = new ArrayList(30);
   readFromFile(file);
}

/**
  * Konstruktor: aus Pfad p Kokszuordnung erzeugen
  * @param p Pfad
  * @param quelle = matrix.holeQuelle();
  * @param info = matrix.holeInfo();
  */
public KoksZuordnung(Pfad p, String quelle, String info){
   zuordnungen = new ArrayList(30);
   String s = p.toWagnerString(quelle, info);
   readFromString(s);
}


/** Den ganzen Info kram mal in ein Interface/Klasse verpacken, das nutzen viele Klassen!*/
/** Info-String liefern */
public String holeInfo(){
   return info;
}

/** Quelle liefern*/
public String holeQuelle(){
   return quelle;
}



/**
  * liest Daten aus einer Datei im Wagner-Format ein und liefert ein entsprechendes Objekt
  * @param file Dateiname als String
  */
private void readFromFile(String file){
      try{
         BufferedReader in = new BufferedReader(new FileReader(file));
         readFromBufferedReader(in);
      }catch(FileNotFoundException e){
         System.err.println("IO-Fehler. (file not found)");
      }
}

/**
  * liest Daten im WagnerFormat aus einem String ein
  */
private void readFromString(String s){
      BufferedReader in = new BufferedReader(new StringReader(s) );
      readFromBufferedReader(in);
}



/**
  * Hier wird die eigentliche Einlesarbeit gemacht...
  */
private void readFromBufferedReader(BufferedReader in){
   int segment;
   String lang1;
```

```
String lang2;

StringBuffer l1,l2;

Character komma = new Character(',');

try{

   quelle = in.readLine();       // Quelle und
   info = in.readLine();         // Info lesen


   String line = in.readLine();  //

while (line != null){
   StringTokenizer tok = new StringTokenizer(line, "\t \n", false);

   // Segmentnummer holen
   segment = Integer.parseInt(tok.nextToken());

   // neue Zurodnung anlegen
   Zuordnung z = new Zuordnung();

   // Liste der Sätze der ersten Sprache holen
   l1 = new StringBuffer();
   l1.append(tok.nextToken());
   // Solange am Ende des Puffers ein "," rumhängt
   while(new Character(l1.charAt(l1.length()-1)).toString().equals(",")){
      l1.append(tok.nextToken());
   }
   lang1 = l1.toString();     // nur für die Ausgabe...
   // l1 in die Satznummern zerlegen

   if(new Character(l1.charAt(0)).compareTo(new Character('[')) == 0){
      l1.deleteCharAt(0);

      while(new Character(l1.charAt(0)).compareTo(new Character(']')) != 0){
         StringBuffer number = new StringBuffer();
         while(Character.isDigit(l1.charAt(0))){
            number.append(l1.charAt(0));
            l1.deleteCharAt(0);
         }

         // Lang1-Satzzahl in z aufnehmen
         z.addL1(Integer.parseInt(number.toString()));

         if(new Character(l1.charAt(0)).compareTo(new Character(',')) == 0)
            l1.deleteCharAt(0);

      }
   }


   // Liste der Sätze der zweiten Sprache holen
   l2 = new StringBuffer();
   l2.append(tok.nextToken());
   // Solange am Ende des Puffers ein "," rumhängt
   while(new Character(l2.charAt(l2.length()-1)).toString().equals(",")){
      l2.append(tok.nextToken());
   }
   lang2 = l2.toString();
```

```
                if(new Character(l2.charAt(0)).compareTo(new Character('[')) == 0){
            l2.deleteCharAt(0);

            while(new Character(l2.charAt(0)).compareTo(new Character(']')) != 0){
               StringBuffer number = new StringBuffer();
               while(Character.isDigit(l2.charAt(0))){
                  number.append(l2.charAt(0));
                  l2.deleteCharAt(0);
               }
               // Lang2-Satzzahl in z aufnehmen
               z.addL2(Integer.parseInt(number.toString()));

               if(new Character(l2.charAt(0)).compareTo(new Character(',')) == 0)
                  l2.deleteCharAt(0);

            }
         }

         // So, die Zuordnung dieses Segmentes ist fertig eingelesen.
         zuordnungen.add(z);
         line = in.readLine();
      }


      }catch(IOException e){
         System.err.println(e.getMessage());
         System.err.println("KoksZuordnung.readFromBufferedReader: IO-Fehler beim lesen aus BufferedReader "+
      }


   }

   /** Erzeugt Wagner-Format (Datei, nicht Pizza) */
   public String toString(){

      StringBuffer buf = new StringBuffer();

      for(int i = 0; i < zuordnungen.size(); i++){
         Zuordnung z = (Zuordnung)zuordnungen.get(i);

         buf.append(z.toString());
         buf.append("\n");
      }

      return buf.toString();

   }



   /**
    *  Liefert die Liste der Koordinaten, die im Alignment liegen
    *  x,y,x1,y1,x2,y2,...xn,yn
    */
   public ArrayList betroffeneKoordinaten(){
      ArrayList koord = new ArrayList();
      for(int i = 0; i < zuordnungen.size(); i++){
         Zuordnung z = (Zuordnung)zuordnungen.get(i);

         z.betroffeneKoordinaten(koord);
```

```
      }
      return koord;
   }

   /**
     * Zum Test mal die betroffenen Koordinaten ausgeben
     */
   public void printKoord(){
      ArrayList x = betroffeneKoordinaten();

      for(int i=0; i < x.size(); i+=2){
         System.out.println("("+x.get(i)+","+x.get(i+1)+")");
      }
   }



   /**
     * Zuordnungen rausrücken
     * (Wird benutzt um HTML-Tabelle in HTMLKocher zu bauen...)
     */
   public ArrayList getZuordnungen(){
      return zuordnungen;
   }


   /** winziger Test: Lese Matrix aus Datei args[0] ein und gebe sie auf System.out aus. */
   public static void main (String[] args){
      // Wagner-Datei einlesen und KoksZuordnung daraus kochen
      KoksZuordnung k = new KoksZuordnung(args[0]);

      // Zuordnung ausgeben
      System.out.println(k);

      // Diejenigen Koordinaten ausgeben, die von der Zuordnung betroffen sind
      k.printKoord();   // Die können dann in Mavis eingezeichnet werden...
   }


}
```

M.5.4 de.denkselbst.matrix.Matrix.MatrixView.java

```
package de.denkselbst.matrix;

import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrix.Matrix;
import de.denkselbst.beobachtung.Beobachter;
import de.denkselbst.mavis.ShowHTMLFrame;

import java.io.*;
import java.net.URL;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JViewport;
```

```
import java.awt.geom.Point2D;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Image;
import java.awt.Toolkit;
import java.util.ArrayList;
import java.util.Iterator;
// für Table-Model
import javax.swing.table.AbstractTableModel;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.DefaultCellEditor;
import javax.swing.table.TableCellRenderer; import javax.swing.JLabel;
import javax.swing.JDialog;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JColorChooser;
import javax.swing.BorderFactory;
import javax.swing.border.Border; import javax.swing.JScrollPane;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import java.awt.*;
import java.awt.event.*;
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.TableColumn;
import javax.swing.ListSelectionModel;

/**
  *  Diese Klasse stellt eine Matrix grafisch dar und stellt Fenster für
  *  die Alignmentpfade bereit. Fast alles
  *  sichtbare in Mavis stammt von hier.
  */

public class MatrixView extends JPanel implements Beobachter, MouseMotionListener, MouseListener{

   /** Referenz auf die anzuzeigende Matrix */
   private Matrix matrix;
   /** Liste von verschiedenen Alignmentpfaden(/KoksZuordnungen) als Elem-Objekte verpackt..*/
   private ArrayList pfade;

   /** Innere Klasse: Bündelt Pfad/KoksZuordnung mit Farbe, Info und Sichtbarkeitsstatus */
   public class Elem{
      Object object;
      boolean visible;
      Color color;
      String info;

      /** Konstruktor */
      public Elem(Object o, boolean vis, Color c, String i){
         object = o;
         visible = vis;
         color = c;
         info = i;
      }
   }
```

```
/** Referenz auf Objekt innerer Klasse T, das die Pfade einer JTable anzeigbar macht*/
T tableModel;
/** zugehörige JTable*/
JTable table;

/** Abmessungen der gezeichneten Matrixfelder */
private int breite, hoehe;
/** Hält Hintergrundbild für besseres Aussehen, wenn keine Matrix vorhanden ist (nur Spielerei...)*/
private Image hgBild;

/** Flagge: Diagonale zeichnen oder nicht */
private boolean zeigeDiag = false;

/** Referenz auf Fenster, dass zu Matrixzellen Werte anzeigt*/
private XYInfo xyinfo;
/** Zeigt wenn möglich tatsächliche Sätze an (X,Y) an (wenn Korpus aus Datei stammt...)*/
private SatzInfoDialog satzinfo;
/** Zeigt wenn möglich Abstandinfo an (X,Y) an */
private AbstandInfoDialog abstandinfo;
/** Dialog mit weiteren Infos zur Matrix*/
private MatrixStatInfoDialog matrixInfo;

/**
  * legt neue MatrixView ohne Inhalt an
  */
public MatrixView(){
   matrix = null;                // keine Matrix
   pfade = new ArrayList();            // Liste anlegen
   breite = 10;                  // Feldbreite
   hoehe =  10;               // Feldhöhe
   setSize(690,450);             // Standartgröße
   setPreferredSize(new Dimension(690,450));       //  "


//////////
// Fenster zur Anzeige einiger statistischer Informationen
   matrixInfo = new MatrixStatInfoDialog(matrix);
   matrixInfo.setLocation(0,400);
   matrixInfo.show();

//////////
// x,y-Koordinaten, Farbe und Wert werden in diesem Fenster angezeigt.
   xyinfo = new XYInfo();
   xyinfo.setLocation(0,100);
   addMouseMotionListener(this);    // Dazu benötigen wir die Mauskoordinaten
   addMouseListener(this);

//////////
// SatzInfoDialog
   satzinfo = new SatzInfoDialog(matrix);
   satzinfo.setLocation(0,430);

//////////
// AbstandInfoDialog
   abstandinfo = new AbstandInfoDialog(matrix);
   abstandinfo.setLocation(0,500);

//////////
// Innere Klasse, die als TableModel dient, in gang bringen
   tableModel= new T(this);
```

```
    JDialog f = new JDialog((JFrame)null,"Mavis - Alignments");

    // Dieser Dialog soll nicht schliessbar sein...
    f.setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);

    table = new JTable(tableModel);
    table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

         //Set up renderer and editor for the Favorite Color column.
          tableModel.setUpColorRenderer(table);
    tableModel.setUpColorEditor(table);

    // Spaltenbreiten setzen
    TableColumn column = null;
    column = table.getColumnModel().getColumn(0);
    column.setPreferredWidth(25);
    column = table.getColumnModel().getColumn(1);
    column.setPreferredWidth(25);

    JScrollPane scrollPane = new JScrollPane(table);
    table.setPreferredScrollableViewportSize(new Dimension(220, 70));

    f.getContentPane().add(scrollPane);
    f.setLocation(0,200);
    f.pack();
    f.show();
//
//////////
}

/**
  * legt neue MatrixView mit Inhalt an
  * @param m Matrix, die angezeigt werden soll
  */
public MatrixView(Matrix m){
    // Defaultkonstruktor laufen lassen
    this();
    // Matrix merken
    matrix = m;
    // Wir sind an allen Änderungen an der Matrix interessiert!
    m.beobachterAnmelden(this);
    // bevorzugte Bildschirmabmessungen setzen
    setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));
    //setPreferredSize(new Dimension(640,400));
    // neu zeichnen!
    repaint();
}



/**
  * neues Matrix Objekt einpflanzen
  */
public void setzeMatrix(Matrix m){
    if(matrix != null) matrix.beobachterAbmelden(this);   // wir melden uns als Beobachter der alten Matrix
    matrix = m;
    if(matrix != null){
       matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
       // Maße setzen
       setSize(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe);
```

```
      setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));
   }

   // bei Gelegenheit müssen wir uns neu zeichnen
   repaint();

   // häßlich! matrixInfo bescheid sagen
   matrixInfo.setzeMatrix(m);
   // nochmal ganz häßlich:
   satzinfo.setzeMatrix(m);
   abstandinfo.setzeMatrix(m);

}


/**
  * einen Pfad in die Liste der Alignment-Pfade aufnehmen,
  * Info-String ist mitzuliefern!
  */
public void fuegeAlignmentPfadHinzu(Pfad p, String info){
   if(pfade==null) pfade = new ArrayList();

   // Pfad als Element verpacken
   Elem e = new Elem(p, true, new Color((int)(Math.random()*127)+128,15,255-(int)(Math.random()*127)) , info)

   // Element in die PfadListe aufnehmen
   pfade.add(e);

   // Tabelle anstupsen
   tableModel.updateTable();

   // Selber neu zeichnen
   repaint();
}

/**
  * den selektierten Pfad aus der Liste der Alignment-Pfade löschen
  */
public void entferneSelektiertenAlignmentPfad(){
   if(pfade!=null){
      int index = table.getSelectedRow();
      if(index != -1){
         pfade.remove(index);
         tableModel.updateTable();
         repaint();
      }
   }
}


/**
  * den selektierten Pfad aus der Liste der Alignment-Pfade als HTML anzeigen
  */
public void zeigeSelektiertenAlignmentPfadAlsHTML(){
   if(pfade!=null){
      int index = table.getSelectedRow();
      if(index != -1){

         HTMLKocher kocher = new HTMLKocher(); // erzeugt html
         Elem elem = (Elem)pfade.get(index);   // selektiertes Element besorgen
         Object o = elem.object;               // enthaltenes Objekt rausholen
```

```
            // Ist das Objekt in der Liste ein Pfad?
            // Dann konvertieren!
            if(o instanceof Pfad){
               KoksZuordnung kneu = new KoksZuordnung((Pfad) o, matrix.holeQuelle(), matrix.holeInfo());
               o = kneu;
            }

            // ist es (jetzt) eine KoksZuordnung?
            // Dann mittels ShowHTMLFrame Ausgeben!
            if(o instanceof KoksZuordnung){
               new ShowHTMLFrame("Alignment",""+kocher.kocheTabelle((KoksZuordnung)o,satzinfo.getShowTagsObje
            }

         }
      }
   }

   public boolean istEtwasSelektiert(){
      return (table.getSelectedRow() != -1);
   }

   /**
     * Liste der Alignment-Pfade komplett löschen
     */
   public void loescheAllePfade(){
      pfade = null;
      repaint();
      // Tabelle anstupsen
      tableModel.updateTable();
   }

   /**
     * Schreibt den selektierten Pfad (so vorhanden) in die Datei file
     */
   public void speichereSelektiertenAlignmentPfad(File file){

      int index = table.getSelectedRow();
      if(index != -1){
         Elem elem = (Elem)pfade.get(index);

         if(elem.object instanceof Pfad){
            Pfad pfad = (Pfad) elem.object;

            try{
               FileWriter fw = new FileWriter(file);
               fw.write(pfad.toWagnerString(matrix.holeQuelle(), matrix.holeInfo()+" :: Alignment erzeugt von
               fw.close();

            }catch(IOException e){
               System.err.println("Fehler beim Schreiben der Datei: "+ e.getMessage());
            }

         }
      }

   }


   /**
     * KoksZuordnung in die Liste der Pfade aufnehmen
```

```
  */
public void fuegeKoksZuordnungEin(KoksZuordnung zugekokst, String info){
   if(zugekokst==null) return;
   if(pfade==null) pfade = new ArrayList();

   Elem e = new Elem(zugekokst, true, new Color(15,(int)(Math.random()*255),(int)(Math.random()*255)) , info)
   pfade.add(e);

   // Tabelle anstupsen
   tableModel.updateTable();
   repaint();
}




/**
  * In dieser Methode finden das Zeichnen der
  * Matrix und der Pfade statt.
  */
public void paint(Graphics g){

   // erstmal den Hintergrund leerputzen
   super.paintComponent(g); //paint background

   // gefüllte Matrixfelder zeichnen
   if(matrix != null){
      for(int i=0; i< matrix.holeBreite(); i++){
         for(int j=0; j<matrix.holeHoehe(); j++){
            g.setColor(getColor(matrix.holeDatumVonStelle(i,j)));
            g.fillRect(i*(breite), j*(hoehe), breite, hoehe);
         }
      }
   }

   // Diagonale zeichnen
   if(zeigeDiag && matrix != null){
      g.setColor(new Color(0,0,0));
      g.drawLine(0,0,matrix.holeBreite()*breite, matrix.holeHoehe()*hoehe);
   }


   // KoksZuordnung zeichnen falls vorhanden (z.B. um Church und Gale aus Datei anzuzeigen...)
   // Pfadliste abklappern und sichtbare KoksZuordnungen ausgeben...
   if(pfade != null){
      for(int p = 0; p < pfade.size(); p++){
         // Element besorgen
         Elem elem = (Elem)pfade.get(p);
         // Handelt es sich um eine sichtbare KoksZuordnung, dann zeichnen
         if( (elem.object instanceof KoksZuordnung) && elem.visible){
            g.setColor(elem.color);
            KoksZuordnung kokszu = (KoksZuordnung) elem.object;
            ArrayList kokskoord = kokszu.betroffeneKoordinaten();
            int x,y;
            for (int i = 0; i < kokskoord.size(); i+=2){
               x = ((Integer)kokskoord.get(i)).intValue();
               y = ((Integer)kokskoord.get(i+1)).intValue();

               if( (x >= 0) && (x < matrix.holeBreite()) &&
```

```
                    (y >= 0) && (y < matrix.holeHoehe()) ){
                  g.drawRect(x*(breite)+1, y*(hoehe)+1, breite-2, hoehe-2);
               }

            }
         }
      }
   }// if(pfade!=null)

   if(matrix == null){
      // keine Matrix, dann mit tollem Logo protzen!
      // Logo ggbf. laden
      if(hgBild == null){
         URL url = this.getClass().getResource("/de/denkselbst/mavis/logo.gif");
         Toolkit toolkit = Toolkit.getDefaultToolkit();
         hgBild = toolkit.getImage(url);
      }

      int h = hgBild.getHeight(this);
      int w = hgBild.getWidth(this);

      g.drawImage(hgBild, 0, 0, new Color(0,0,0), this);
   }



   // Pfade kennzeichnen
   if (pfade != null){                        // es gibt mindestens einen Pfad

      // Pfade abarbeiten
      for (int n=0; n < pfade.size(); n++){

         // Elem besorgen
         Elem elem = (Elem)pfade.get(n);
         // Handelt es sich um einen sichtbaren Pfad, dann zeichnen
         if( (elem.object instanceof Pfad) && elem.visible){
            // Farbe setzen
            g.setColor(elem.color);

            // Pfad besorgen
            Pfad p = (Pfad)elem.object;
            // Iterator über die Knoten des Pfades besorgen
            Iterator it = p.iterator();

            int oldx=0;       // Startpunkt des Linienzuges
            int oldy=0;       //  "

            int b=breite/2 ;//+(n*2);  // Mittelpunkt-Offset in die gezeichneten Matrixfelder
            int h=hoehe/2 +(n*2);   //  "

            while(it.hasNext()){
               Pfad.Koordinate k = (Pfad.Koordinate) it.next();
               g.drawLine((oldx*breite)+b,(oldy*hoehe)+h,(k.x*breite)+b,(k.y*hoehe)+h);
               oldx=k.x;
               oldy=k.y;
            }
         }

      }
   }
}
```

```
   /**
     * einen double Wert zwischen 0 und 1 gegen einen Grünton eintauschen: 0 = sehr helles Grün,
     * 1 = sehr dunkles Grün, man könnte auch Schwarz dazu sagen
     */
   public Color getColor(double wert){
      //if(wert < 0.0000000000000000000) return new Color(150,0,0);
      //if(wert > 1.0000000000000000000) return new Color(0,0,150);
      float w = (float) wert;
      if (w>=1.0f) w=1.0f;
      if (wert > 1.0) return new Color(150,5,5);
      return new Color(0,1-w,0); //
   }

   /**
     *  Blockgröße setzen (Zeichnen)
     */
   public void setzeBlockGroesse(int breite, int hoehe){
      if (matrix == null) return;   // Defensiv...

      this.breite = breite;
      this.hoehe = hoehe;

      // Maße setzen
      setSize(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe);
      setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));
      revalidate();

      repaint();
   }


   /**
     *  Schalter: Diagonale zeichnen oder nicht? (Zeichnen)
     */
   public void zeigeDiagonale(boolean z){
      zeigeDiag = z;
      repaint();
   }

   public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
      // (neue) Maße setzen
      setSize(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe);
      setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));

      revalidate();

      repaint();
   }



////////////////////////////////////////////
////////////////////////////////////////////
/** MouseMotionListener und MouseListener     */
//
   public void mouseDragged(MouseEvent e){
      // nichts tun...

   }
```

```
   public void mouseMoved(MouseEvent e){

      if(matrix != null && xyinfo != null){

         int x=0 ,y=0;

         x+= e.getX();
         y+= e.getY();

         x = x/ breite;
         y = y/ hoehe;

         if(x >= matrix.holeBreite() || y >= matrix.holeHoehe() ){
            xyinfo.dead();
            if (satzinfo != null) satzinfo.dead();
            if (abstandinfo != null) abstandinfo.dead();
         }else{
            double data = matrix.holeDatumVonStelle(x,y);
            xyinfo.update(x,y,getColor(data),data);
            if(satzinfo!=null) satzinfo.show(x,y);
            if(abstandinfo!=null) abstandinfo.show(x,y);
         }

      }

   }


   public void mouseClicked(MouseEvent e){}
   public void mouseEntered(MouseEvent e){}
   public void mouseExited(MouseEvent e){//xyinfo.dead(); if(satzinfo!=null)satzinfo.dead();
   }
   public void mousePressed(MouseEvent e){}
   public void mouseReleased(MouseEvent e){}



////////////////////////////////////////////
////////////////////////////////////////////
//
/** Memberklasse, die in einem eigenen Fenster
 *  die aktuellen MatrixKoordinaten und den zugehörigen Wert anzeigt
 */
   public class XYInfo extends JDialog  {

      JLabel xp,yp,col,val;

      /** Konstruktor */
      public XYInfo(){
         super((JFrame)null,"Matrixzellen - Info");
         setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);

         xp = new JLabel("x"); //xp.setHorizontalAlignment(JLabel.RIGHT);
         yp = new JLabel("y");
         col = new JLabel("        ");
         col.setOpaque(true);
         val = new JLabel("        ");
         JPanel p = new JPanel(new GridLayout(4,2));

         p.add(new JLabel("X")); p.add(xp);
         p.add(new JLabel("Y")); p.add(yp);
```

```
         p.add(new JLabel("Farbe")); p.add(col);
         p.add(new JLabel("Wert")); p.add(val);

         getContentPane().add(p);
         setSize(new Dimension(220,70));
         show();

      }

      /** Die neuen Werte zur Anzeige bringen */
      public void update(int x, int y, Color color, double value){
         xp.setText(""+x);
         yp.setText(""+y);
         col.setBackground(color);
         col.setForeground(color);
         val.setText(""+value);
         //repaint(); // verursachen die Labels das von selbst?
      }

      /** Teilt diesem Objekt mit, dass keine anzeigbaren Werte vorliegen.
        * z.B. Maus ausserhalb des Fensters
        */
      public void dead(){
         xp.setText("-");
         yp.setText("-");
         val.setText("-");
         col.setBackground(Color.lightGray);
         col.setForeground(Color.lightGray);
      }

   }



////////////////////////////////////////////
////////////////////////////////////////////
//
/** eklige Memberklasse, die als Tablemodel dient */
//


public class T extends AbstractTableModel{

   String[] columnNames = new String[]{"sichtbar","Farbe","Info","Ende der Durchsage"};
   private MatrixView mv;

   public T(MatrixView mv){
      this.mv = mv;
   }

   public void updateTable(){
      fireTableDataChanged();
   }

   public int getColumnCount(){
      return 3;
   }

   public int getRowCount(){
      if(pfade!=null) return pfade.size();   // aus der umgebenden Klasse die ArrayListe Pfade
      else return 0;
```

```
   }

   public String getColumnName(int col) {
         return columnNames[col];
   }

   public Class getColumnClass(int c) {
      //return getValueAt(0, c).getClass();

      if(c==0) return new Boolean(true).getClass();
      if(c==1) return new Color(1,2,3).getClass();
      if(c==2) return new String("").getClass();

      return null;
   }


   public Object getValueAt(int row, int column){
      if(row < pfade.size()){
         Elem elem = (Elem) pfade.get(row);
         if(column==0) return new Boolean(elem.visible);
         else if(column==1) return elem.color;
         else if(column==2) return elem.info;
      }
      else{
         if(column==0) return new Boolean(false);
         if(column==1) return new Color(1,2,3);
         if(column==2) return new String("dummy");
      }
      return new Integer(666);
   }

   public boolean isCellEditable(int row, int col) {
      if ( (col >= 0) && (col <=1) )
         return true;
      return false;
   }


   public void setValueAt(Object value, int row, int col) {
      Elem elem = (Elem)pfade.get(row);

      if(col==0){
         elem.visible=((Boolean)value).booleanValue();
      }
      else if(col==1){
         elem.color=(Color)value;
      }

      fireTableCellUpdated(row, col);
      mv.repaint();
   }

//////////////////////////
/* noch eine innere Klasse, diesmal, um
 * Farben in der Pfadliste darzustellen und
 * Farbauswahl zu ermöglichen. Quelle: Java-Tutorial von Sun.
 */

    class ColorRenderer extends JLabel
                        implements TableCellRenderer {
```

```
    Border unselectedBorder = null;
    Border selectedBorder = null;
    boolean isBordered = true;

    public ColorRenderer(boolean isBordered) {
        super();
        this.isBordered = isBordered;
        setOpaque(true); //MUST do this for background to show up.
    }

public Component getTableCellRendererComponent(
                            JTable table, Object color,
                            boolean isSelected, boolean hasFocus,
                            int row, int column) {
        setBackground((Color)color);
        if (isBordered) {
            if (isSelected) {
                if (selectedBorder == null) {
                    selectedBorder = BorderFactory.createMatteBorder(2,5,2,5,
                                              table.getSelectionBackground());
                }
                setBorder(selectedBorder);
            } else {
                if (unselectedBorder == null) {
                    unselectedBorder = BorderFactory.createMatteBorder(2,5,2,5,
                                              table.getBackground());
                }
                setBorder(unselectedBorder);
            }
        }
        return this;
    }
}

private void setUpColorRenderer(JTable table) {
    table.setDefaultRenderer(Color.class,
                             new ColorRenderer(true));
}     //Set up the editor for the Color cells.

private void setUpColorEditor(JTable table) {
    //First, set up the button that brings up the dialog.
    final JButton button = new JButton("") {
        public void setText(String s) {
            //Button never shows text -- only color.
        }
    };
    button.setBackground(Color.white);
    button.setBorderPainted(false);
    button.setMargin(new Insets(0,0,0,0));         //Now create an editor to encapsulate the button, and
    //set it up as the editor for all Color cells.
    final ColorEditor colorEditor = new ColorEditor(button);
    table.setDefaultEditor(Color.class, colorEditor);         //Set up the dialog that the button brings up.
    final JColorChooser colorChooser = new JColorChooser();
    ActionListener okListener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            colorEditor.currentColor = colorChooser.getColor();
        }
    };
    final JDialog dialog = JColorChooser.createDialog(button,
                                    "Pick a Color",
                                    true,
```

```
                                    colorChooser,
                                    okListener,
                                    null); //XXXDoublecheck this is OK         //Here's the code that bri
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                button.setBackground(colorEditor.currentColor);
                colorChooser.setColor(colorEditor.currentColor);
                //Without the following line, the dialog comes up
                //in the middle of the screen.
                //dialog.setLocationRelativeTo(button);
                dialog.show();
            }
        });
    }     /*
     * The editor button that brings up the dialog.
     * We extend DefaultCellEditor for convenience,
     * even though it mean we have to create a dummy
     * check box.  Another approach would be to copy
     * the implementation of TableCellEditor methods
     * from the source code for DefaultCellEditor.
     */
    class ColorEditor extends DefaultCellEditor {
        Color currentColor = null;         public ColorEditor(JButton b) {
                super(new JCheckBox()); //Unfortunately, the constructor
                                        //expects a check box, combo box,
                                        //or text field.
            editorComponent = b;
            setClickCountToStart(1); //This is usually 1 or 2.             //Must do this so that editing sto
            b.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    fireEditingStopped();
                }
            });
        }         protected void fireEditingStopped() {
            super.fireEditingStopped();
        }         public Object getCellEditorValue() {
            return currentColor;
        }         public Component getTableCellEditorComponent(JTable table,
                                                    Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column) {
            ((JButton)editorComponent).setText(value.toString());
            currentColor = (Color)value;
            return editorComponent;
        }
    }

}

}
```

M.5.5 de.denkselbst.matrix.Matrix.MatrixHistogramView.java

```
package de.denkselbst.matrix;

import de.denkselbst.beobachtung.Beobachter;

import javax.swing.JPanel;
```

```
import java.awt.Color;
import java.awt.Graphics;

/**
  * Diese Klasse ist eine grafische Komponente, die in einem eigenen
  * Fenster ein Histogramm eines dem Konstruktor übergebenen
  * Matrix-Objektes zeichnet.
  */

public class MatrixHistogramView extends JPanel implements Beobachter {

   /** Referenz auf die als Histogramm darzustellende Matrix */
   private Matrix matrix;
   private int[] histogramm;
   private double schrittGroesse;

   int viewbreite = 180;
   int viewhoehe = 70;
   float scale = 1.0f;     // wird benutzt, um die Balken immer schön groß und deutlich zu zeichnen...

   /** Konstruktor */
   public MatrixHistogramView(Matrix m){
      matrix = m;
      schrittGroesse = 0.05;
      // Wir sind an allen Änderungen an der Matrix interessiert!
      matrix.beobachterAnmelden(this);
      histogramm = berechneHistogramm(schrittGroesse);
      computeScale();
   }


   /** Das tatsächliche Zeichnen findet hier statt. */
   public void paint(Graphics g){
      // Anzahl Matrixfelder berechnen
      int groesse = matrix.holeHoehe() * matrix.holeBreite();

      int breite = viewbreite -20 - (histogramm.length - 1); // Gesamtbreite für alle Balken zusammen
      int balkenBreite = breite / histogramm.length ;        // Breite eines Balkens

      // erstmal den Hintergrund leerputzen
      super.paintComponent(g); //paint background

      for(int i = 0; i < histogramm.length; i++){
         float col = (float) schrittGroesse * (i+1);
         if (col < 0 || col > 1){
             g.setColor(new Color(175,10,10));
         }
         else g.setColor(new Color(0,1-col,0));

         int laenge = (int) (scale * (( (1.0 * histogramm[i]) / groesse) * viewhoehe));
         g.fillRect(10+i+ i*balkenBreite, viewhoehe- laenge, balkenBreite, laenge);
      }
   }

   /** Histogramm / Balkenhöhen /-Anzahlen ausrechnen
         * und Balkenhöhenarray liefern.
         */
   private synchronized int[] berechneHistogramm (double stepsize){
      int hoehe = matrix.holeHoehe();
      int breite = matrix.holeBreite();
```

```
      int anzahlBalken = (int)Math.ceil(1.0 / stepsize);
      int [] balken = new int[anzahlBalken];

      for(int i = 0; i < breite; i++){
         for(int j = 0; j < hoehe; j++){

            double wert = matrix.holeDatumVonStelle(i,j);
            for(int z = 0; z < anzahlBalken; z++){
               if (wert >= z * stepsize && wert < (z+1)*stepsize){
                  balken[z]++;
                  continue;
               }
            }
         }
      }
      return balken;
   }


   /** Berechnet internen Vergrößerungsfaktor, damit das
          * Histogramm immer schön sichtbar ist.
          */
   private void computeScale(){
      if(histogramm == null) return;

      int groesse = matrix.holeHoehe() * matrix.holeBreite();
      int max = 0;
      for(int i=0; i<histogramm.length; i++){
         if (histogramm[i] > max) max = histogramm[i];
      }

      if(max ==0) max = 1;

      float relmax = (float) (1.0f*max / groesse);
      scale = 1.0f / relmax;

      if (scale == 0) scale = 1.0f;
   }




   /** Schrittgrösse verändern*/
   public void setzeSchrittgroesse(double g){
      schrittGroesse = g;
      histogramm = berechneHistogramm(schrittGroesse);
      computeScale();
      repaint();
   }


   /** Beobachter: Ein update der Anzeige wird nötig. */
   public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
      histogramm = berechneHistogramm(schrittGroesse);
      computeScale();
      repaint();
   }

   /** Die Beobachtung des Matrix-Objektes einstellen */
   public void abmelden(){
      matrix.beobachterAbmelden(this);
```

```
   }

}
```

M.5.6 de.denkselbst.matrix.Matrix.MatrixStatInfoDialog.java

```
package de.denkselbst.matrix;

import de.denkselbst.beobachtung.Beobachter;
import javax.swing.*;
import java.awt.*;

/** Diese Klasse zeigt in einem eigenen Fenster
  *?Dateiname, Mittelwert, Varianz, Standartabweichung der
  * dem Konstruktor übergebenen Matrix an.
  */
   // Alle diese Views bei Gelegenheit ausgliedern!
   // Vielleicht im nächsten Projekt.


   public class MatrixStatInfoDialog extends JDialog implements Beobachter{

      Matrix matrix;
      JLabel dateiname, info;
      JLabel mittelwert, varianz, standardabweichung;
      JLabel dimensionen, minimum, maximum;

      /** Konstruktor */
      public MatrixStatInfoDialog(Matrix mat){
         super((JFrame)null,"Matrixinfo");
         setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);

         this.setzeMatrix(mat);

         JLabel dn = new JLabel("Quelle");
         JLabel nf = new JLabel("Info");
         JLabel di = new JLabel("x * y");
         JLabel mi = new JLabel("Mittelw.");
         JLabel va = new JLabel("Varianz");
         JLabel st = new JLabel("Stdabw.");
         JLabel min = new JLabel("Minimum");
         JLabel max = new JLabel("Maximum");

         dateiname = new JLabel("...");
         info = new JLabel("...");
         mittelwert = new JLabel("...");
         varianz = new JLabel("...");
         standardabweichung = new JLabel("...");
         dimensionen = new JLabel("...");
         minimum = new JLabel("...");
         maximum = new JLabel("...");

         JPanel p = new JPanel(new GridLayout(8,2));

         p.add(dn); p.add(dateiname);
         p.add(nf); p.add(info);
         p.add(di); p.add(dimensionen);
         p.add(mi); p.add(mittelwert);
         p.add(va); p.add(varianz);
         p.add(st); p.add(standardabweichung);
```

```
         p.add(min); p.add(minimum);
         p.add(max); p.add(maximum);

         getContentPane().add(p);
         setSize(new Dimension(220,100));

         update();

         show();
      }

      /** Wird durch Observermuster angestossen, sobald
                  * das Matrixobjekt sich ändert. Sorgt dafür, das
                  * die angezeigten Werte aktualisiert werden.
                  */
      private void update(){
         if(matrix != null){
            dateiname.setText(matrix.holeQuelle());
            info.setText(matrix.holeInfo());
            dimensionen.setText(""+matrix.holeBreite()+" x "+matrix.holeHoehe());
            mittelwert.setText(""+matrix.berechneMittelwert());
            varianz.setText(""+matrix.berechneVarianz());
            standardabweichung.setText(""+matrix.berechneStandardabweichung());
            minimum.setText(""+matrix.wasIstDerKleinsteWert());
            maximum.setText(""+matrix.wasIstDerGroessteWert());
         }

      }


      /**
       * neues Matrix Objekt einpflanzen
        */
      public void setzeMatrix(Matrix m){
         if(matrix != null) matrix.beobachterAbmelden(this);   // wir melden uns als Beobachter der alten Mat
         matrix = m;
         if(matrix != null){
            matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
            update();
         }
      }

      public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
         update();
      }
   }
```

M.5.7 de.denkselbst.matrix.Matrix.SatzInfoDialog.java

```
package de.denkselbst.matrix;

//import de.denkselbst.matrix.Matrix;

import de.denkselbst.beobachtung.Beobachter;
import javax.swing.*;
import java.awt.*;

/**
  * Diese Klasse zeigt in einem Fenster Sätze textuell an, die zu der
  * aktuell selektierten Matrixzelle gehören.
```

```
  * Dazu wird dem Konstruktor ein Matrix-Objekt übergeben. SatzInfoDialog
  * meldet sich bei der Matrix als Beobachter an und besorgt sich die Quell-
  * Information derselben. Es wird ein ShowTags-Objekt angelegt, das mittels
  * der Quelleninformation möglicherweise die entsprechenden Textdateien
  * lesen kann.
  * Das MatrixView-Objekt reagiert auf Mausbewegungen und weist sein
  * SatzInfoDialog-Objekt an, die zur aktuell unter dem Mauskursor
  * befindlichen Matrixzelle gehörenden Sätze anzuzeigen.
  */
public class SatzInfoDialog extends JDialog implements Beobachter{

      private JTextArea de, en;
      private ShowTags textquelle;
      private Matrix matrix;

      /** Konstruktor */
      public SatzInfoDialog(Matrix mat){
         super((JFrame)null,"Satzinfo");
         setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
         matrix = mat;
         de = new JTextArea("deutscher Text"/*,8,35*/);
         en = new JTextArea("englischer Text"/*,8,35*/);
         de.setLineWrap(true); de.setWrapStyleWord(true);
         en.setLineWrap(true); en.setWrapStyleWord(true);
         JPanel p = new JPanel(new GridLayout(1,2));
         JScrollPane ensp = new JScrollPane(en, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SC
         en.setPreferredSize(new Dimension(320,100));
         JScrollPane desp = new JScrollPane(de, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SC
         de.setPreferredSize(new Dimension(320,100));
         p.add(ensp);
         p.add(desp);
         getContentPane().add(p);
         setSize(640,120);
         updateQuelle();
         pack();
         show();
      }

      /** Liefert benutztes ShowTags-Objekt */
      public ShowTags getShowTagsObject(){
         return textquelle;
      }

      /** Erzeugt ShowTags-Objekt für internen Gebrauch */
      public void updateQuelle(){
         if(matrix != null)
            textquelle = new ShowTags(matrix.holeQuelle());
         else textquelle = null;
      }

      /** Sorgt dafür, das die Sätze x und y textuell im
                  * Fenster angezeigt werden
                  */
      public void show(int x, int y){
         if(textquelle == null) return;
         de.setText(textquelle.getY(y));
         en.setText(textquelle.getX(x));
      }

      /** Wird aufgerufen, wenn der Mauszeiger nicht über
                  * einer Matrixzelle steht. In diesem Fall kann nichts
```

```
                * angezeigt werden.
                */
      public void dead(){
         de.setText("...");
         en.setText("...");
      }

      /**
       * neues Matrix Objekt einpflanzen
        */
      public void setzeMatrix(Matrix m){
         if(matrix != null) matrix.beobachterAbmelden(this);   // wir melden uns als Beobachter der alten Mat
         matrix = m;
         if(matrix != null){
            matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
            updateQuelle();
         }
      }

      public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
         updateQuelle();

      }

}
```

M.5.8 de.denkselbst.matrix.Matrix.ShowTags.java

```
package de.denkselbst.matrix;

import java.io.*;
import java.util.ArrayList;

/**
  * Diese Klasse besorgt, falls möglich, die zu einer Matrix
  * gehörenden Texte und liefert auf Anfrage Sätze (Strings) zu
  * übergebenen Satznummern (int).
  */
public class ShowTags{

   private ArrayList deutsch;
   private ArrayList englisch;

   /** Konstruktor */
   public ShowTags(String quelle){
      //nur wenn quelle das Format: "Dokument-Datei ../korpus/de-news/1997/05/de-news-1997-05-03"
      //hat, können wir etwas damit anfangen!
      String dateistamm = null;
      if(quelle.startsWith("Dokument-Datei")){  // Dann brauchbar!
         dateistamm = quelle.substring(15);
      }

      deutsch = new ArrayList();
      englisch = new ArrayList();

      fillFromFile(deutsch,dateistamm+".de.tag");
      fillFromFile(englisch,dateistamm+".en.tag");
```

```
   System.err.println("de: "+deutsch.size()+" Sätze");
   System.err.println("en: "+englisch.size()+" Sätze");
}

// x englisch
// y deutsch
/** Liefert Satz x aus L1 als String. */
public String getX(int x){
   if (x >= 0 && x < englisch.size()) return ((String)englisch.get(x)).trim();
   else return "keine Daten an x="+x;
}
/** Liefert Satz y aus L2 als String. */
public String getY(int y){
   if (y >= 0 && y < deutsch.size()) return ((String)deutsch.get(y)).trim();
   else return "keine Daten an y="+y;
}

/* Liefert die Anzahl der Sätze aus L1.*/
public int xmax(){
   return englisch.size();
}
/* Liefert die Anzahl der Sätze aus L2.*/
public int ymax(){
   return deutsch.size();
}

/**
  * Füllt übergebene ArrayList <SATZ>-Weise aus Datei dateiname
  */
private void fillFromFile(ArrayList list, String dateiname) {

   // Nicht gerade der beste Weg um herauszufinden, dass kein vernünftiger Dateiname!
   if(dateiname.equals("null.de.tag") || dateiname.equals("null.en.tag")){
      System.err.println("diese Quelle kann ich nicht anzeigen");
      return;
   }

   //System.err.println("****** Showtags: "+dateiname);

   try{
      String line = null;
      StringBuffer akkumulator = new StringBuffer();
      String wort = null;

      int zustand = 0;



      File file = new File(dateiname);
      BufferedReader br = new BufferedReader(new FileReader(file));

      while ( (line=br.readLine()) != null ){

         int pos = line.indexOf("\t");
         if(pos >= 0 && pos < line.length()) wort = line.substring(0,pos).trim();
         else wort = line;
         //System.out.println(wort);

         //               (pos >=0 : Indikator dafür, dass es sich um Inhalt handelt)
         if(zustand == 0 && pos > 0  ){
         //System.err.println("[1] akk: "+akkumulator.toString()+" wort: "+wort);
```

```
               // Inhalt aufsammeln
               akkumulator.append(wort+" ");
            }else if (zustand == 0 && (line.equals("<ABSATZ>") || line.equals("<SATZ>") || line.equals("<segm
               list.add(akkumulator.toString());
               akkumulator = new StringBuffer();
               zustand = 1;
            }else if(zustand == 1 && pos >= 0  ){
               zustand = 0;
            //System.err.println("[2] akk: "+akkumulator.toString()+" wort: "+wort);
               akkumulator.append(wort+" ");
            }else if (zustand == 1 && (line.equals("<ABSATZ>") || line.equals("<SATZ>") || line.equals("<segm
               // Wir bleiben in diesem Zustand

            }

         }//eowhile

         // Ist noch was im Akkumulator? Dann anhängen
         // Kann passieren, wenn abschließende Delimiter fehlen
         if(akkumulator.length()>0){
            list.add(akkumulator.toString());
         }



      }catch(IOException e){
         System.out.println(e.getMessage());
      }

   }




   /**
     * Kleines Main zu Test- / Demozwecken
     */
   public static void main (String[] args) {
      ShowTags st = new ShowTags(args[0]);

      // Teste ShowTags in Verbindung mit HTMLKocher

      String x = args[0].substring(15);
      System.err.println("öffne: "+x+".align");
      KoksZuordnung k = new KoksZuordnung(x+".align");

      HTMLKocher kocher = new HTMLKocher();
      System.out.println(""+kocher.kocheTabelle(k,st,"fakealigner","keine Parameter"));
   }

}
```

M.5.9 de.denkselbst.matrix.Matrix.HTMLKocher.java

```
package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.Date;
```

```
/**
  * Diese Klasse enthält eine statische Methode, die
  * KoksZuordnung unter Zuhilfenahme eines ShowTags-Objektes
  * und diversen info-Strings ls HTML-String liefert.
  * So kann ein Alignment in einer HTML-View textuell
  * dargestellt werden.
  */

public class HTMLKocher{

   /**
     * Diese statische Methode gibt eine KoksZuordnung @param k
     * unter Zuhilfenahme eines ShowTags-Objektes @param s
     * und diversen info-Strings
     * als HTML-String zurück.
     * @param info Info-String des Aligners / Matrix (z.B. AStern, Lmed)
     * @param parameter String, der die gesetzten Parameter des Aligners beschreiben soll (in Zukunft)
     * @param quelle Herkunft der Daten (analog zu Matrix.quelle)
     *
     */
   public static String kocheTabelle(KoksZuordnung k, ShowTags s, String alinfo, String parameter){

   String info =  k.holeInfo();
   String quelle = k.holeQuelle();

   // alternierende Hintergrundfarben  für die Tabelle
   String[] colors = new String[]{"BGCOLOR=\"#FFFFCC\"", "BGCOLOR=\"#FFFFE4\""};
   int currentColorIndex = 0;
   String currentColor;

      ArrayList zuordnungen = k.getZuordnungen();

      StringBuffer buf = new StringBuffer("<html>\n <head></head>\n  <body>");
      // Später noch alle verfügbaren Infos zum verwendeten Aligner
      // (Art, Parameter!)
      // und ein gif der Tabelle
      // ausgeben, damit Ausdrucke besser verglichen werden können.

      buf.append("\n Datum: "+new Date(System.currentTimeMillis())+" <br> Quelle: "+quelle+" <br> Info:"+info+"<
      buf.append("<h1>Alignment: "+alinfo+"</h1>");
      buf.append("\n  <table border cols = 2>");

      // Hintergrundfarbe für Tabellenheader-Zeile besorgen
      currentColor = colors[currentColorIndex];
      // nächste Farbe (Index ins Array bestimmen)
      currentColorIndex = (currentColorIndex + 1) % colors.length;
      // HTML-Tabelle: Headerzeile ausgeben
      buf.append("\n  <tr "+currentColor+" ><th>L1</th><th>L2</th></tr>");

      // Alle Zuordnungen abarbeiten und ausgeben
      for(int i = 0; i < zuordnungen.size(); i++){
         KoksZuordnung.Zuordnung z = (KoksZuordnung.Zuordnung)zuordnungen.get(i);

         // z ist ein Segment
         ArrayList lang1 = z.getL1();  // Sprache 1 (Satz)Nummern
         ArrayList lang2 = z.getL2();  // Sprache 2 (Satz)Nummern

         // Hintergrundfarbe für die Tabellenzeile besorgen
         currentColor = colors[currentColorIndex];

         // HTML-Tabelle: Zeile mit Hintergrundfarbe öffnen
```

```
         buf.append("\n   <tr "+currentColor+" >\n    <td valign=top>");

         // Sprache 1 behandeln:
         // Segmentnummer und Text ausgeben
         for(int j = 0; j < lang1.size(); j++){
            buf.append("["+((Integer)lang1.get(j)).intValue()+"]:");    // Segmentnummer
            buf.append( s.getY(((Integer)lang1.get(j)).intValue()) );   // Text
            buf.append("<br>\n");                                       // Linebreak
         }

         // HTML-Tabelle: nächste Spalte aufmachen
         buf.append("    </td>\n    <td valign=top>");

         // Sprache 2 behandeln
         // Segmentnummer und Text ausgeben
         for(int j = 0; j < lang2.size(); j++){
            buf.append("["+((Integer)lang2.get(j)).intValue()+"]:");    // Segmentnummer
            buf.append( s.getX(((Integer)lang2.get(j)).intValue()) );   // Text
            buf.append("<br>\n");                                       // Linebreak
         }

         // HTML-Tabelle: Spalte, Zeile schließen
         buf.append("</td>\n  </tr>");

         // nächste Farbe (Index ins Array bestimmen)
         currentColorIndex = (currentColorIndex + 1) % colors.length;

      }
      buf.append("\n  </table>\n </body>\n</html>");

      return buf.toString();
   }
}
```

M.5.10 de.denkselbst.matrix.Matrix.AbstandInfoDialog.java

```
package de.denkselbst.matrix;

//import de.denkselbst.matrix.Matrix;

import de.denkselbst.beobachtung.Beobachter;
import javax.swing.*;
import java.awt.*;

/**
  * Nach dem Vorbild von SatzInfoDialog geklonte Klasse zum
  * Anzeigen von Debugging-Informationen für Abstandsmaße.
  * Autor: Joachim.
  */

public class AbstandInfoDialog extends JDialog implements Beobachter{

      private JTextArea de, en;
      private ShowAbstand textquelle;
      private Matrix matrix;

      public AbstandInfoDialog(Matrix mat){
         super((JFrame)null,"Abstandinfo");
         setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
```

```
   matrix = mat;
   de = new JTextArea("Info 1"/*,8,35*/);

   de.setLineWrap(true); de.setWrapStyleWord(true);

   JPanel p = new JPanel(new GridLayout(1,1));

   JScrollPane desp = new JScrollPane(de, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SC
   de.setPreferredSize(new Dimension(320,100));

   p.add(desp);

   getContentPane().add(p);

   setSize(640,120);

   //setLocation(200,550);

   updateQuelle();

   pack();
   show();
}

public ShowAbstand getShowAbstandObject(){
   return textquelle;
}


public void updateQuelle(){
   if(matrix != null)
      textquelle = new ShowAbstand(matrix.holeQuelle());
   else textquelle = null;
}

public void show(int x, int y){
   if(textquelle == null) return;
   de.setText(textquelle.getAbstandinfo(x,y));
}

public void dead(){
   de.setText("...");
}



/**
 * neues Matrix Objekt einpflanzen
  */
public void setzeMatrix(Matrix m){
   if(matrix != null) matrix.beobachterAbmelden(this);   // wir melden uns als Beobachter der alten Matrix
   matrix = m;
   if(matrix != null){
      matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
      updateQuelle();
   }
}

public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
   //System.out.println("..."+nachricht);
   updateQuelle();
```

```
      }



   public static void main(String[] args){
      AbstandInfoDialog s = new AbstandInfoDialog(null);
   }

}
```

M.5.11 de.denkselbst.matrix.Matrix.ShowAbstand.java

```
package de.denkselbst.matrix;

import java.io.*;
import java.util.ArrayList;
import java.util.StringTokenizer;

/**
  * Nach dem Vorbild von ShowTags geklonte Klasse zum
  * Anzeigen von Debugging-Informationen für Abstandsmaße.
  * Autor: Joachim.
  */


public class ShowAbstand{



   private ArrayList abstandinfo;

        private int breite, hoehe;

   /** Konstruktor */
   public ShowAbstand(String quelle){

      //nur wenn quelle das Format: "Dokument-Datei ../korpus/de-news/1997/05/de-news-1997-05-03"
      //hat, können wir etwas damit anfangen!
      String dateistamm = null;
      if(quelle.startsWith("Dokument-Datei")){  // Dann brauchbar!
         dateistamm = quelle.substring(15);
      }


      abstandinfo = new ArrayList();

      //System.err.println("dateistamm: "+dateistamm);

      fillFromFile(abstandinfo,dateistamm+".abstandinfo");


      System.err.println(":"+abstandinfo.size()+" Abstandinfos");
   /*
      for(int i = abstandinfo.size()-5; i < abstandinfo.size(); i++){
         System.err.println("de:"+i+": "+getAbstandinfo(i));
      }
      System.err.println();
```

```
*/


   }

   // x englisch
   // y deutsch

   public String getAbstandinfo(int x, int ny){
               int y = breite * ny + x;
      if (y >= 0 && y < abstandinfo.size()) return ((String)abstandinfo.get(y)).trim();
      else return "keine Daten an x,y = "+x+", "+ny;
   }


   public int xmax(){
      return breite * hoehe;
   }

   public int ymax(){
      return breite * hoehe;
   }


   /**
     * Füllt übergebene ArrayList <SATZ>-Weise aus Datei dateiname
     */
   private void fillFromFile(ArrayList list, String dateiname) {

      // Nicht gerade der beste Weg um herauszufinden, dass kein vernünftiger Dateiname!
      if(dateiname.equals("null.de.abstandsinfo") || dateiname.equals("null.en.abstandsinfo")){
         System.err.println("diese Quelle kann ich nicht anzeigen");
         return;
      }


      try{
         String line = null;
         StringBuffer akkumulator = new StringBuffer();

         int zustand = 0;

         File file = new File(dateiname);
         BufferedReader br = new BufferedReader(new FileReader(file));

                        line = br.readLine();
                        StringTokenizer tok = new StringTokenizer(line, "\t \n", false);

                        // nächstes Token besorgen
                        if(!tok.hasMoreTokens()){
                                while(!tok.hasMoreTokens() && line!=null){
                                        line = br.readLine();
                                        tok = new StringTokenizer(line, "\t \n", false);
                                }
                        }
                        // breite der Matrix holen
                        breite = Integer.parseInt(tok.nextToken());

                        // nächstes Token besorgen
                        if(!tok.hasMoreTokens()){
                                while(!tok.hasMoreTokens() && line!=null){
```

```
                                        line = br.readLine();
                                        tok = new StringTokenizer(line, "\t \n", false);
                                }
                        }
                        // Höhe der Matrix holen
                        hoehe = Integer.parseInt(tok.nextToken());

        System.err.println("ShowAbstand breite = " + breite + ", hoehe = " + hoehe + "\n");

        line = null;

        while ( (line=br.readLine()) != null ){

           //System.out.println(line);

           if (zustand == 0 && (line.equals("<ABSATZ>") || line.equals("<SATZ>") || line.equals("<segmentgre
              list.add(akkumulator.toString());
              akkumulator = new StringBuffer();
              zustand = 1;
           }else if(zustand == 0){
              // Inhalt aufsammeln
              akkumulator.append(line+"\n");
           }else if (zustand == 1 && (line.equals("<ABSATZ>") || line.equals("<SATZ>") || line.equals("<segm
              // Wir bleiben in diesem Zustand
           }else if(zustand == 1){
              zustand = 0;
              akkumulator.append(line+"\n");
           }


        }//eowhile

        // Ist noch was im Akkumulator? Dann anhängen
        // Kann passieren, wenn abschließende Delimiter fehlen
        if(akkumulator.length()>0){
           list.add(akkumulator.toString());
        }



     }catch(IOException e){
        System.out.println(e.getMessage());
     }

  }




  /**
    * kleines Main
    */
  public static void main (String[] args) {

     ShowAbstand st = new ShowAbstand(args[0]);
/* // Teste ShowAbstand, indem alles ausgegeben wird
     for(int i = 0; i < st.ymax(); i++){
        System.out.println("de:"+i+": "+st.getAbstandinfo(i));
     }
```

```
      System.out.println();

*/

      // Teste ShowAbstand in Verbindung mit HTMLKocher

      String x = args[0].substring(15);
      System.err.println("öffne: "+x+".align");
      KoksZuordnung k = new KoksZuordnung(x+".align");

      // HTMLKocher kocher = new HTMLKocher();
      // System.out.println(""+kocher.kocheTabelle(k,st,"fakealigner","keine Parameter"));  // st kein ShowTags

   }


}
```

M.5.12 de.denkselbst.matrix.Matrix.WriteRandomMatrix.java

```
package de.denkselbst.matrix;

/**
  *   Write random matrix to stdout. Values are between 0.0 and 1.0. First command line
  *   argument is taken as xdim, second as ydim. Default dims: 10 x 10
  *   Data is written row per row from left to right.
  *   The output has the following format: xdim \t ydim \t val1 \t ... \t valn
  *     and can be read by Matrix-objects.
  *   The aim is to provide test material to Matrix-objects.
  */
public class WriteRandomMatrix{
   public static void main (String[] args){
      int x = 10; // default dimensions
      int y = 10; //  "

      if (args.length > 0)
         x = Integer.parseInt(args[0]);
      if (args.length > 1)
         y = Integer.parseInt(args[1]);

      System.out.println("random matrix");
      System.out.println("fuer Testzwecke");
      System.out.print(""+x+"\t"+y);

      for(int i = 0; i < x; i++){
         for(int j = 0; j < y; j++){
            System.out.print("\t"+Math.random());
         }
      }
   }
}
```

M.5.13 de.denkselbst.matrix.Matrix.Convert.java

```
package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.StringTokenizer;
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.FileReader;
import java.io.File;
import java.io.FileWriter;

/**
  * Die Main-Methode dieser Klasse verannlasst, das eine zeilenweise
  * eingelesene Matrix spaltenweise ausgegeben wird.
  * inputdatei = argument[n], outputdateiname = inputdateiname+".converted"
  *
  * Da es nach einigen Anlaufschwierigkeiten mit dem Matrixformat keine
  * Probleme mehr gab, war es schon früh nicht mehr nötig, Matrizen mit
  * diesem Programm zu drehen.
  *
  * !!Diese Klasse ist daher obsolet.!!
  * Allerdings ist sie noch immer Teil des CVS-Baumes. Der Code zum
  * Einlesen einer Matrix wurde hier dupliziert,
  * was eine Sünde ist.
  */
public class Convert{
   public static void main (String[] args) throws IOException {
      for(int a = 0 ; a < args.length; a++){
         double m[][] = readFromFile(args[a]);
         int x = m.length;
         int y = m[0].length;
         File outputFile = new File(args[a]+".converted");
         FileWriter out = new FileWriter(outputFile);

         out.write(""+x+"\t"+y+"\t");
         for(int i = 0 ; i < x; i++){
            for(int j = 0; j < y; j++){
               out.write(""+m[i][j]+"\t");
            }
         }
         out.close();
      }
   }


   /**
     * liest Daten aus einer Datei ein und liefert double[][]
     * @param file Dateiname als String
     */
   private static double[][] readFromFile(String file){
      int x = 0;                              // Abmessungen der Matrix
      int y = 0;                              //  "
      double[][] matrix = new double[0][0];  // Referenz auf Matrix

      try{
         BufferedReader in = new BufferedReader(new FileReader(file));
         String line = in.readLine();
         StringTokenizer tok = new StringTokenizer(line, "\t \n", false);

         // nächstes Token besorgen
         if(!tok.hasMoreTokens()){
            while(!tok.hasMoreTokens() && line!=null){
               line = in.readLine();
               tok = new StringTokenizer(line, "\t \n", false);
            }
         }
         // breite der Matrix holen
```

```
         x = Integer.parseInt(tok.nextToken());

         // nächstes Token besorgen
         if(!tok.hasMoreTokens()){
            while(!tok.hasMoreTokens() && line!=null){
               line = in.readLine();
               tok = new StringTokenizer(line, "\t \n", false);
            }
         }
         // Höhe der Matrix holen
         y = Integer.parseInt(tok.nextToken());

         // Matrix anlegen
         matrix = new double[x][y];
         String token = null;

         // Zeilenweise einlesen
         for (int j = 0; j < y; j++){
            for (int i = 0 ; i < x ; i++){
               // Nächstes Token besorgen
               if(!tok.hasMoreTokens()){
                  while(!tok.hasMoreTokens() && line!=null){
                     line = in.readLine();
                     tok = new StringTokenizer(line, "\t \n", false);
                  }
               }
               token = tok.nextToken();
               matrix[i][j] = Double.parseDouble(token);
            }
         }
      }catch(IOException e){
         System.err.println(e.getMessage());
         System.err.println("Fehler beim Lesen der Datei "+file+" . Liefere leere 6x6 Matrix.");
         return new double[6][6];
      }

      return matrix; // Die Matrix liefern
   }



}
```

M.5.14 de.denkselbst.matrixAligner.MatrixAligner.java

```
package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.beobachtung.Beobachter;

/**
Diese Klasse ist Superklasse aller Klassen, die einen Alignment-Pfad für eine Matrix
ermitteln wollen, können oder sollen.

Die Klasse MatrixAligner schreibt vor, wie sich Objekte zu verhalten haben,
die einen Alignmentpfad in einer Matrix ermitteln wollen. Für das GUI-Tool
Mavis wurden Methoden bereitgestellt, das Alignment in einem separaten Thread
auszuführen. Für Kommandozeilentools wird diese Funktionalität nicht benötigt.
Alignments werden in dieser Version als Pfad-Objekte abgeliefert. Später in
```

```
der Entwicklung wurde die Klasse KoksZuordnung geschaffen, die Alignments
adequater repräsentiert. Pfade lassen sich in jedem Fall in KoksZuordnungen
umwandeln.


Soll das Alignment in einem eigenen Thread ablaufen, so ist die Methode
public void align(AlignmentObserver obs) aufzurufen. Dabei ist zu beachten, dass
dieser Aufruf pro Alignerobjekt nur einmal stattfinden darf, da sonst der zweite Aufruf
den ersten Observer überschreibt. Also: Alignerobjekt instantiieren, einmal alignen, dann
Alignerobjekt wegwerfen. Anschließend neues Alignerobjekt machen, alignen, wegwerfen, usw.

*/

public abstract class MatrixAligner implements Beobachter, AlignmentObservable, Runnable{
////////////////////////////////////////////////////////////////////
//
// Konstruktor, Matrix merken, Auskunft über alignerIdentString geben, Konfigurationsdialog aufrufen
//
   /** enthält Referenz auf die Matrix */
   Matrix matrix;
   /** String, der den Typ dieses Aligners beinhaltet (muss/sollte in den Unterklassen gesetzt werden) */
   String alignerIdentString;

   /**
     * erzeugt ein MatrixAligner Objekt mit einer gegebenen Matrix
     * @param matrix die Matrix
     */
   public MatrixAligner(Matrix matrix){
      this.matrix = matrix;
      if (this.matrix != null)
         this.matrix.beobachterAnmelden(this);  // wir melden uns als Beobachter der Matrix an
      this.alignerIdentString = "abstract MatrixAligner";
   }

   /**
     * setzt eine neue Matrix ein
     * @param m die Matrix
     */
   public void setzeMatrix(Matrix m){
      if (this.matrix != null)
         this.matrix.beobachterAbmelden(this);  // wir melden uns als Beobachter der alten Matrix ab
      this.matrix = m;
      if (this.matrix != null)
         this.matrix.beobachterAnmelden(this);  // wir melden uns als Beobachter der neuen Matrix an
   }


   /**
     * Liefert den alignerIdentString. Der kann dann z.B. im GUI als Fenstertitel gesetzt werden.
     */
   public String alignerInfo(){
      return alignerIdentString;
   }

   /**
     * Soll in Unterklassen so überschrieben werden, dass
     * ein modaler Konfigurationsdialog abgefahren wird und die Einstellungen
     * dann gestzt werden.
     * Bietet der Aligner keine Einstellmöglichkeiten, ist nichts weiter zu tun.
     * Diese Methode wird von der grafischen Oberfläche (Mavis/MACV) aufgerufen, nachdem
     * der Aligner instanttiert wurde und bevor das Alignment gestartet wurde.
```

```
     */
   public void konfigurationsDialogAbfahren(){
      System.err.println("("+alignerIdentString+" bietet keine Einstellmöglichkeiten.)");
   }

////////////////////////////////////////////////////////////////////
//
// Alignment, Observer-Behandlung
//
   /** für Benachrichtigung des Observers, falls in einem Extrathread gearbeitet wird */
   protected AlignmentObserver observer = null;
   /** Flagge, die Abbruch des Alignmentvorganges signalisiert. Wird in der Regel in der
     * Hauptschleife der Methode abgefragt, die das Alignment berechnet               */
   protected boolean cancelled = false;
   /** Flagge, die auf true gesetzt wird, wenn der Alignmentvorgang beendet ist.
       Vor einem erneuten Alignen ist diese Flagge wieder auf false zu setzen         */
   protected boolean finished = false;
   /** Flagge, die auf true gesetzt wird, wenn der Alignmentvorgang beginnt.*/
   protected boolean started = false;

   /**
     * ermittelt (den) Alignment-Pfad
     * (ist von den Unterklassen zu implementieren)
     */
   public abstract Pfad align();

   /**
     * Diese Methode startet das Alignment in einem neuen Thread.
     * Das Alignment-Ergebnis
     * wird per Observer-Mechanismus an den AlignmentObserver obs zugestellt.
     * (aus interface AlignmentObservable)
     */
   public void align(AlignmentObserver obs){
      observer = obs;                                                          // Auftraggeber merken
      System.err.println(alignerIdentString+": In neuem Thread gestartet.");   // Nachricht ausgeben
      Thread al = new Thread(this);                                            // neuen Thread anlegen
      al.start();                                                              // und starten
   }

   /**
     * Diese Methode setzt die Flagge zum Beenden des Alignments
     * (aus interface AlignmentObservable)
     */
   public void cancel(){
      cancelled = true;                                                        // Flagge setzen
      System.err.println(alignerIdentString+": Alignmentvorgang abgebrochen"); // Nachricht ausgeben
      abmelden();                                                              // Observer/MatrixBeobachtung v
   }

   /**
     * Meldet den MatrixAligner als Beobachter von der Matrix ab und vergisst
     * seinen Observer.
     * "abmelden" sollte das GUI zum MatrixAligner sagen, bevor es ihn entsorgt,
     * da die Matrix den MatrixAligner sonst noch als Beobachter registriert hat,
     * es also noch eine Referenz gibt und der Aligner folglich nicht
     * vom Garbage-Collector eingesammelt werden kann!!!
     */
   public void abmelden(){
      matrix.beobachterAbmelden(this);                                         // von der Matrix abmelden
      observer=null;                                                           // Observer vergessen
   }
```

```
   /**
     * run-Methode für Arbeit in einem eigenen Thread
     */
   public void run(){
      align();          // Return-Wert brauchen wir nicht, der wird per ObserverInterface geliefert...
   }



////////////////////////////////////////////////////////////////////
//
// Methode, die das Interface de.denkselbst.beobachtung.Beobachter verlangt
//
   /**
     * Auf Änderungen in der Matrix reagieren die Aligner sehr empfindlich, nämlich
     * mit Abbruch des Alignments.
     * (aus interface Beobachter)
     */
   public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
//System.err.println("s: "+started+"  f: "+finished);
      if(started == true && finished == false ){   // Nur, wenn das Alignen schon begonnen wurde und noch im
         System.err.println("Beobachtetes Objekt ("+o.getClass()+") meldet Änderung:\n   "+nachricht);
         System.err.println("Deshalb bricht sich "+alignerIdentString+" ab.");
         cancel();
         informObserver("Alignment abgebrochen, da die Matrix geändert wurde.",false);
      }
   }



////////////////////////////////////////////////////////////////////
//
// Convenience-Methoden zur Kommunikation mit dem Observer
//
   /** Zähler für informObserverZZZ */
   private int notifyCount;

   /**
     * Convenience-Methode: Status-Nachricht an Observer
     * (prüft auf observer != null)
     */
   protected void informObserver(String text, boolean erfolg){
      if(observer!=null){
         observer.statusInformation(text,erfolg);
      }
   }

   /**
     * Convenience-Methode: Status-Nachricht an Observer, aber nur jede 100.
     * Nachricht wird tatsächlich weitergereicht.
     * (prüft auf observer != null)
     */
   protected void informObserverZZZ(String text, boolean erfolg){
      notifyCount++;
      if(notifyCount > 100){
         if(observer!=null){
            observer.statusInformation(text,erfolg);
         }
         notifyCount = 0;
      }
```

```
   }

   /**
     * Convenience-Methode: Alignmentergebnis an Observer weitergeben
     * (prüft auf observer != null)
     * @param p der Ergebnispfad (bei Mißerfolg leeren Pfad übergeben)
     * @param erfolg wenn false, war das Alignment erfolglos und p
     * enthält nicht den Pfad (sondern ist z.B. leer, o.ä.)
     */
   protected void informObserverAboutFinishedAlignment(Pfad p, boolean erfolg){
      if(observer!=null){
         observer.alignmentFertig(p, erfolg);
      }
   }



}  // Ende der Klassendefinition
```

M.5.15 de.denkselbst.matrixAligner.AlignmentObservable.java

```
package de.denkselbst.matrixAligner;

/**
  * Dieses Interface legt die Methoden fest, die ein Alignment-Objekt beherrschen
  * muss, wenn es Alignment in einem eigenen Thread anbieten will. (Observer-Muster)
  */
public interface AlignmentObservable{
   /** Starte Alignment unter Beobachtung/Einfluss von @param obs*/
   public void align(AlignmentObserver obs);
   /** AlignmentObserver kann durch Aufruf dieser Methode den Alignmentvorgang abbrechen */
   public void cancel();
}
```

M.5.16 de.denkselbst.matrixAligner.AlignmentObserver.java

```
package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Pfad;

/**
  * Es handelt sich um ein Interface, das die Methoden festlegt, die ein Objekt beherrschen
  * muss, wenn es Alignment in einem anderen Thread lostreten will und
  * über den Fortschritt benachrichtigt werden will. (Observer-Muster)
  */
public interface AlignmentObserver{
   /** diese Methode wird vom observierten MatrixAligner-Objekt
            aufgerufen, wenn der Alignmentprozess ein Ende gefunden hat. @param erfolg
       enthält einen boolschen Wert, der andeutet, ob das Unternehmen erfolgreich war, wenn
       ja, dann enthält @param p den Alignmentpfad.
     */
   public void alignmentFertig(Pfad p, boolean erfolg);
   /** Übermittelt Statusinformationen (z.B. zu Ausgabe in einer grafischen Oberfläche) */
   public void statusInformation(String s, boolean erfolg);
}
```

M.5.17 de.denkselbst.matrixAligner.AStarMatrixAligner.java

```
package de.denkselbst.matrixAligner;
```

```
import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

import java.util.ArrayList;
//import java.util.Iterator;

import com.bluemarsh.adt.BinomialHeap;
import com.bluemarsh.adt.BinomialHeap.Node;


// Für den Konfig-Dialog
import javax.swing.JDialog;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JCheckBox;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

/**
  * Diese Klasse ist ein MatrixAligner, der den A*-Suchalgorithmus für die Berechnung des Pfades
  * einsetzt.
  *
  * Dreht man die Suchrichtung um (rechts unten nach links oben), so kann aufgrund der
  * Methode, die die Nachfolgeknoten berechnet kein Ergebnis gefunden werden.
  */

public class AStarMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

////////////////////////////////////////////////////////////////
//
// Konstruktor und Konfigurations-Dialog
//
   /**
     * Konstruktor
     */
   public AStarMatrixAligner(Matrix matrix){
      super(matrix);                                  // Matrix durchreichen an Superklasse
      this.alignerIdentString = "AStarMatrixAligner";    // Beschreibung setzen
   }


   /**
     * modalen Konfigurationsdialog abfahren und Aligner entsprechend konfigurieren.
     */
   public void konfigurationsDialogAbfahren(){
      ConfigDialog conf = new ConfigDialog();
   }



////////////////////////////////////////////////////////////////
//
// Flaggen, die das Verhalten des Aligners steuern
//
   /** Flagge, ob Schritte nach Südwesten erlaubt sind   */
```

```
   protected boolean SW_allowed = false;
   /** Flagge, ob Schritte nach Westen erlaubt sind      */
   protected boolean W_allowed = false;
   /** Flagge, ob Schritte nach Nordwesten erlaubt sind  */
   protected boolean NW_allowed = false;
   /** Flagge, ob Schritte nach Norden erlaubt sind      */
   protected boolean N_allowed = false;
   /** Flagge, ob Schritte nach Nordosten erlaubt sind   */
   protected boolean NE_allowed = false;
   // Schritte nach Osten, Südosten und Süden sind implizit erlaubt!

   /** Wenn möglich, nur Nachfolger mit besserem Wert als Schwelle erzeugen. Findet sich
       kein solcher, sollen alle Nachfolger erzeugt werden
       use_Threshold = Flagge */
   protected boolean use_Threshold = true;
   /** expandThreshold = Schwelle, die die Nachfolger unterbieten müssen */
   protected double expandThreshold = 0.75;

   /** Ist diese Flagge gesetzt, wird der Pfad von unten rechts nach oben links gesucht, statt
       wie defaultmäßig von oben rechts nach unten links */
   protected boolean reverseAlignDir = false;

////////////////////////////////////////////////////////////7
//
// Implementation des A*-Aligners
//
   BinomialHeap open;   // open - Menge von Knoten als Heap
   Node[] nodes;        // Tabelle aller Knoten (dient u.a. als closed)

   // innere Klasse für Knoten
   private class Node{
      int nodeID;
      BinomialHeap.Node heapNodeRef;
      Node cheapestAncestor;
      ArrayList successors;
      boolean open;
      boolean closed;
      double g;
      double h;
      int xpos;
      int ypos;

      Node(int id, int x, int y){
         nodeID = id;
         xpos = x;
         ypos = y;
         successors = new ArrayList(3);
         // restlichen Werte automatisch 0, bzw. null
      }

      double getF(){
         return g+h;
      }

      // Alle bekannten Nachfolger von neuen g-Kosten in Kenntnis setzen; zx, zy = Zielkoordinaten
      // Enthält möglicherweise Fehler!
      // Wird aber derzeit offenbar nicht aufgerufen...
      void update(BinomialHeap openHeap, int zx, int zy){
//System.err.println("updating");
         for(int i = 0; i < successors.size(); i++){
            int succID = ((Integer)successors.get(i)).intValue();    // ID besorgen und damit
```

```
            Node succNode = nodes[succID];                          // Knoten besorgen


            // Hat Nachfolger schlechteren f-Wert als f'=unser g+seine Kosten+ sein h?
            double gSucc = g + matrix.holeDatumVonStelle(succNode.xpos, succNode.ypos);
            double hSucc = matrix.minKost(succNode.xpos, succNode.ypos, zx, zy);
            double newF =  gSucc + hSucc;

            if( newF < succNode.getF() ){
//System.err.println("newF < succNode.getF()");
               succNode.cheapestAncestor = this;      // dann auf uns verweisen
               succNode.g = gSucc;                    // neue g-Kosten eintragen
               succNode.h = hSucc;                    // neue h-Kosten eintragen
               succNode.update(openHeap, zx, zy);     // Änderung propagieren

               // Wenn wir in open sind, muss der heap geändert werden!!!
               if(open){
//System.err.println("we are in open, so we change our value in the heap");
                  openHeap.delete(heapNodeRef);    // alten Eintrag wegwerfen
                  heapNodeRef = openHeap.insert(new Integer(nodeID), getF()); // Knoten mit upgedateten f-Kos
               }
            }//if besserer Wert
         }//for
      }//update

   }//Node




   /**
     * a*-Suche von (x1,y1) nach (x2,y2)
     */
   private Pfad aStarSearch(int x1, int y1, int x2, int y2){

      int schleifenZaehler = 0;
      informObserver("Initialisiere",true);

      // -- Initialisierung allgemein -----------------------------------------------------------
      int breite = matrix.holeBreite();         // Breite und Hoehe der Matrix
      int hoehe = matrix.holeHoehe();           // merken

      open = new BinomialHeap();                // Heap anlegen

      nodes = new Node[breite * hoehe];         // Tabelle für Knoten-Objekte anlegen
      for(int j = 0; j < hoehe; j++){           // und initialisieren
         for(int i = 0; i < breite; i++){
            int id = j*breite + i;              // Knoten fortlaufend durchnummerieren
            Node node = new Node(id,i,j);             // Knoten-Objekt erzeugen
            node.h = matrix.minKost(i, j, x2, y2);    // h setzen
            nodes[id] =  node;                        // und in die Tabelle eintragen
         }
      }

      // -- Initialisierung des open-Heaps ------------------------------------------------------
      int startNodeID = y1 * breite + x1;                     // ID des Startknoten
      Node startNode = nodes[startNodeID];                    // zugehöriges Knoten-Objekt besorgen
      startNode.open = true;                                  // Flagge setzen: Dieser Knoten ist in open
      startNode.g = matrix.holeDatumVonStelle(x1, y1);        // g-Kosten setzen
      //startNode.h = matrix.minKost(x1, y1, x2, y2);         // h-Kosten setzen
      startNode.heapNodeRef = open.insert(new Integer(startNodeID), startNode.getF()); // ID des Startknotens
```

```
      // -- ------------------------------------------------------------------------

      while(true){

         schleifenZaehler++;  // Lediglich ein unwichtiger Zähler, um dem Observer einen Fortschritt melden zu k
         informObserverZZZ(""+schleifenZaehler+". Iteration, "+open.size()+" Knoten sind in open.",true);


         if(open.empty()){    // Wenn open leer ist, ist die Suche gescheitert
            informObserver("Agenda leer in "+schleifenZaehler+". Iteration. Misserfolg, Abbruch.",false);
            informObserverAboutFinishedAlignment(new Pfad(), false);
            return new Pfad();
         }

         // Teste, ob cancel-Nachricht vom observer kam.
         if(cancelled) {
            informObserver("Alignment abgebrochen.",false);
            informObserverAboutFinishedAlignment(new Pfad(), false);
         }


         // besten Knoten aus open entfernen
         int bestNodeID = ((Integer)(open.removeMin().data)).intValue();   // ID des besten Knotens aus open bes
         Node bestNode = nodes[bestNodeID];  // Knoten per ID besorgen
         bestNode.open = false;              // open-Flag auf false setzen
         bestNode.closed = true;             // closed-Flag auf true setzen, Knoten ist jetzt in closed

         // Prüfen, ob bestNode der Zielknoten ist
         if(bestNode.xpos == x2 && bestNode.ypos == y2){
            Pfad pfad = new Pfad();                // Pfad-Objekt
            pfad.haengeAn(x2,y2);                  // Zielknoten anhängen
            Node n = bestNode.cheapestAncestor;    // jeweils Vorgänger anhängen
            while(n != null){
               pfad.haengeAn(n.xpos, n.ypos);
               n = n.cheapestAncestor;
            }
            pfad.umdrehen();                       // Pfad umdrehen
            informObserver("Alignment abgeschlossen in "+schleifenZaehler+". Iteration. "+open.size()+" Knoten s
            informObserverAboutFinishedAlignment(pfad, true);
            return pfad;                           // und liefern
         }
         else{
//System.err.println("Nachfolger erzeugen");
            // Nachfolger von bestNode erzeugen
            ArrayList successors = nachfolger(bestNode.xpos, bestNode.ypos);

            // und abarbeiten
            for(int s = 0; s < successors.size(); s++){
//System.err.println("  arbeite Nachfolger "+(s+1)+"/"+successors.size()+" ab.");
               int succID = ((Integer)successors.get(s)).intValue();    // ID besorgen und damit
               Node succNode = nodes[succID];                           // Knoten besorgen
               double gSucc = bestNode.g + matrix.holeDatumVonStelle(succNode.xpos, succNode.ypos);   // g-Koste

               // Prüfen, ob dieser Nachfolger schon in open bekannt ist
               if(succNode.open){
//System.err.println("    Nachf in open");
                  // Wenn wir diesmal billiger zu diesem Nachfolger gekommen sind:
                  if(gSucc < succNode.g){
//System.err.println("      diesmal billiger: "+gSucc+" < "+succNode.g+" bisheriges f: "+succNode.getF());
                     open.delete(succNode.heapNodeRef);     // alten Eintrag wegwerfen
```

```
                    succNode.cheapestAncestor = bestNode;  // Rückverweis eintragen
                    succNode.g = gSucc;                    // neue g-Kosten eintragen
                    //succNode.h = matrix.minKost(succNode.xpos, succNode.ypos, x2, y2);       // neue h-Kos
                    succNode.heapNodeRef = open.insert(new Integer(succID), succNode.getF());  // Knoten mit
//System.err.println("      neues f: "+succNode.getF());
                 }
              }
              else if(succNode.closed){  // sonst prüfen, ob dieser Nachfolger in closed enthalten ist
//System.err.println("    Nachf in closed");
                 // Liste der Nachfolgeknoten von bestNode um diesen Nachfolger erweitern
                 if(!((ArrayList)bestNode.successors).contains(new Integer(succNode.nodeID)))  // ist dieser
                    ((ArrayList)bestNode.successors).add(new Integer(succNode.nodeID));


                 // Wenn wir diesmal billiger zu diesem Nachfolger gekommen sind:
                 if(gSucc < succNode.g){
//System.err.println("      diesmal billiger: "+gSucc+" < "+succNode.g+" bisheriges f: "+succNode.getF());
                    succNode.cheapestAncestor = bestNode;  // Rückverweis eintragen
                    succNode.g = gSucc;                    // neue g-Kosten eintragen
                    //succNode.h = matrix.minKost(succNode.xpos, succNode.ypos, x2, y2);       // neue h-Kos
//System.err.println("      calling update");
                    succNode.update(open, x2,y2);          // Änderung propagieren dazu muss der Heap leider

                 }

              }
              else{                        // dieser Nachfolger war weder in open noch in closed
//System.err.println("  Nachf weder in open noch in closed");
                 ((ArrayList)bestNode.successors).add(new Integer(succNode.nodeID));  // diesen Nachfolger i
                 succNode.open = true;                                                // dieser Nachfolger i
                 succNode.g = bestNode.g + matrix.holeDatumVonStelle(succNode.xpos, succNode.ypos);        /
                 //succNode.h = matrix.minKost(succNode.xpos, succNode.ypos, x2, y2); // neue h-Kosten besor
                 succNode.heapNodeRef = open.insert(new Integer(succNode.nodeID), succNode.getF());  // Nach

                 if(succNode.cheapestAncestor == null)
   /*claus?*/        succNode.cheapestAncestor = bestNode;  // Rückverweis eintragen - steht nicht in claus l
              }

           }// for

        }// else von if-Zielknoten


     }


   }


   /**
     * Innere Klasse, die für die Erzeugung der Nachfolgeknoten als
     * Datenstruktur dient
     */
    private class helper{
      public int x;
      public int y;
      public double wert;
      public helper(int x, int y, double wert){
         this.x = x;
         this.y = y;
         this.wert = wert;
```

```
    }
  }


/**
   * Nachfolgeknoten eines Knotens erzeugen, geliefert wird Liste von IDs
   *
   * Momentanes Problem: Wenn zusätzliche Nachfolger (NE, W, ...) erlaubt sind, ist
   * bei Verwendung der Schwelle nicht sichergestellt, dass ein Nachfolger gefunden wird,
   * der weiter in Richtung Ziel führt. Es kann aufgrund zu schlechter Bewertung zu einem
   * Zyklus SE -> NE -> SE kommen, was beim A* dazu führt, dass die Agenda sich leert und
   * kein Weg zum Ziel gefunden werden kann. Behoben werden kann das Problem, in dem
   * sichergestellt wird, dass zumindest einer der Nachfolger näher zum Ziel führt, sprich,
   * einer der drei Standartnachfolger erzeugt wird. (Später)
   */
private ArrayList nachfolger(int x, int y){
    int breite = matrix.holeBreite();         // Breite und Hoehe der Matrix
    int hoehe = matrix.holeHoehe();           // merken

    ArrayList potn = new ArrayList(3);        // Potentielle Nachfolger
    double schwelle = expandThreshold;        // lokale Kopie der globalen Schwelle

    // Im ersten Schritt werden alle potentiellen Nachfolgeknoten aufgesammelt.
    // (Im zweiten Schritt werden evtl. einige davon aussortiert.)
    //
    // Erzeugung der legalen "normalen" Nachfolger
    //
    // schräg rechts runter
    if(x+1 < breite && y+1 < hoehe){
       potn.add(new helper(x+1, y+1, matrix.holeDatumVonStelle(x+1, y+1) ));
    }
    // rechts
    if(x+1 < breite){
       potn.add(new helper(x+1, y, matrix.holeDatumVonStelle(x+1, y) ));
    }
    // runter
    if(y+1 < hoehe){
       potn.add(new helper(x, y+1, matrix.holeDatumVonStelle(x, y+1) ));
    }
    // sind Schritte in "ungewöhliche" Richtungen erlaubt, werden die
    // entsprechenden Nachfolgeknoten erzeugt, falls sie legal sind.
    //
    // Westen
    if(W_allowed && x-1 > 0){
       potn.add(new helper(x-1, y, matrix.holeDatumVonStelle(x-1, y) ));
    }
    // Norden
    if(N_allowed && y-1 > 0){
       potn.add(new helper(x, y-1, matrix.holeDatumVonStelle(x, y-1) ));
    }
    // Nordosten
    if(NE_allowed && x+1 < breite && y-1 > 0){
       potn.add(new helper(x+1, y-1, matrix.holeDatumVonStelle(x+1, y-1) ));
    }
    // Südwesten
    if(SW_allowed && x-1 > 0 && y+1 < hoehe){
       potn.add(new helper(x-1, y+1, matrix.holeDatumVonStelle(x-1, y+1) ));
    }
    // NordWesten
    if(NW_allowed && (x-1 > 0) && (y-1 > 0) ){
       potn.add(new helper(x-1, y-1, matrix.holeDatumVonStelle(x-1, y-1) ));
```

```
   }



   // Aus den potentiellen Nachfolgern nun die raussuchen, die es tatsächlich geschafft haben,
   // also z.B. einen besseren Wert als die angegebene Schwelle haben
   // Soll eine Schwelle benutzt werden, und bleibt die Nachfolgerliste nach einem
   // Schleifendurchlauf leer, so wird die Schwelle nach oben korrigert, bis sie
   // >= 1.0 ist, also ausser Kraft gesetzt wurde.
   ArrayList n; // Liste der Nachfolger
   for(;;){
      n = new ArrayList(potn.size());                              // neue Nachfolgerliste anlegen
      for(int i = 0; i < potn.size(); i++){                        // potentielle Nachfolger abgrasen
         helper node = (helper)potn.get(i);                        // aktueller pot. Nachf.
         if(use_Threshold){                                        // Wenn Schwelle benutzt werden soll
            if(node.wert < schwelle){                              // prüfen ob pot. Nachf. Schwelle unters
               n.add( new Integer(node.x + node.y * breite) );     // und dann gegebenenfalls hinzufügen
            }
         }else{                                                    // Soll keine Schwelle verwendet werden,
            n.add( new Integer(node.x + node.y * breite) );        // pot. Nachf. der Liste der Nachf. hinz
         }

      }
      // Zumindest ein Nachfolger hat es geschafft, also Schleife verlassen
      if(!n.isEmpty()) break; // Hier muss später sichergestellt werden, dass zumindest
                              // ein Nachfolger enthalten ist, der weiter zum Ziel
                              // führt, o.ä., so dass kein Zyklus entsteht, der die
                              // Agenda leer werden lässt.



      // Es soll gar keine Schwelle benutzt werden, also Schleife verlassen
      if(!use_Threshold) break;

      // Ist die Schwelle >= 1.00, dann Schleife verlassen
      if(schwelle >= 1.00) break;

      // Ansonsten gilt: Schwelle soll benutzt werden, aber Nachfolgerliste ist leer,
      // also muss die Schwelle korrigiert werden.
      schwelle = 1.05;  // Vielleicht optional eine stufenweise Erhöhung um jeweils die Hälfte der aktuell


      //System.err.println("Schwelle: "+schwelle); // aktuellen lokalen Schwellwert ausgeben
   }


   if(n.isEmpty()) System.err.println("Konnte keine Nachfolger zu "+x+", "+y+" erzeugen.");
   //else System.err.println("."); // Alive-Zeichen ausgeben

   return n;
}



/**
  * ermittelt (den) Alignment-Pfad
  */
public Pfad align(){
   finished = false;
   cancelled = false;
```

```
      started = true;

      Pfad p;
      if(reverseAlignDir){
         p = aStarSearch(matrix.holeBreite()-1,matrix.holeHoehe()-1,0,0);
      }else p = aStarSearch(0,0,matrix.holeBreite()-1,matrix.holeHoehe()-1);

      finished = true;
      started = false;
      return p;
   }




////////////////////////////////////////
//
// Innere Klasse, die innerhalb einer grafischen Oberfläche
// einen Konfigurationsdialog darstellt und die vom Benutzer
// engestellten Werte in den Aligner einträgt.
//
/**
  * Innere Klasse, die innerhalb einer grafischen Oberfläche
  * einen Konfigurationsdialog darstellt und die vom Benutzer
  * engestellten Werte in den Aligner einträgt.
  */
protected class ConfigDialog extends JDialog implements ActionListener{

      JCheckBox[] additionalDirections = new JCheckBox[5];
      JButton okButton;
      JCheckBox useThresholdCheckBox, alignDirCheckBox;
      JTextField ThresholdTextField;

      /** Konstruktor. Legt den modalen Dialog an */
      public ConfigDialog(){
         // Owner, Title, Modal
         super((java.awt.Frame)null,"A* - Einstellungen",true);

         //JPanel panel = new JPanel(new GridLayout(5,2));
         JPanel panel = new JPanel();
         String[] texts = {"SW","W","NW","N","NE"};
         for(int i = 0; i < 5; i++){
            additionalDirections[i] = new JCheckBox(texts[i]);
            panel.add(additionalDirections[i]);
         }

         JPanel panel1 = new JPanel(new GridLayout(2,0));
         panel1.add(new JLabel("Zusätzlich zu W, SW, S erlaubte Richtungen"));
         panel1.add(panel);

         useThresholdCheckBox = new JCheckBox();
         ThresholdTextField = new JTextField();
         ThresholdTextField.setText("0.75");

         JPanel panel2 = new JPanel(new GridLayout(2,2));
         panel2.add(new JLabel("Benutze Schwelle"));
         panel2.add(useThresholdCheckBox);
         panel2.add(new JLabel("Schwellwert für Nachfolger"));
         panel2.add(ThresholdTextField);
```

```
      JPanel panel3 = new JPanel(new GridLayout(0,2));
      panel3.add(new JLabel("von rechts unten nach links oben alignen"));
      alignDirCheckBox=new JCheckBox();
      panel3.add(alignDirCheckBox);

      okButton = new JButton("OK");
      okButton.addActionListener(this);

      getContentPane().setLayout(new GridLayout(3,0));  // 4,0, wenn panel3 auch Platz haben soll

      getContentPane().add(panel1);
      getContentPane().add(panel2);
      //getContentPane().add(panel3); // Umgekehrte Alignmentrichtungs-Option nicht anzeigen, da kaputt

      getContentPane().add(okButton);


      pack();
      show();
   }

   public void actionPerformed(ActionEvent e){

      // Aha, ok wurde gedrückt...
      if(e.getSource() == okButton){

         // Flaggen für erlaubte Wege setzen
         SW_allowed = ((JCheckBox)additionalDirections[0]).isSelected();
         W_allowed  = ((JCheckBox)additionalDirections[1]).isSelected();
         NW_allowed = ((JCheckBox)additionalDirections[2]).isSelected();
         N_allowed  = ((JCheckBox)additionalDirections[3]).isSelected();
         NE_allowed = ((JCheckBox)additionalDirections[4]).isSelected();

         //reverseAlignDir = alignDirCheckBox.isSelected(); // Funktioniert derzeit nicht richtig, Nachfol

         use_Threshold = useThresholdCheckBox.isSelected();
         // Wenn die Schwelle denn benutzt werden soll, den entsprechenden Wert besorgen
         if(use_Threshold){
            expandThreshold = Double.parseDouble(ThresholdTextField.getText());
            if(expandThreshold < 0 || expandThreshold > 1.0) expandThreshold = 0.75;   // Im Zweifelsfall
            // Das Textfeld entweder zu Typchecking verdonnern oder durch einen Slider ersetzen...
         }

         // Einstellungen ausgeben
         System.err.println("A* - Einstellungen:");
         System.err.println("SW_allowed = "+ SW_allowed);
         System.err.println("W_allowed = "+ W_allowed);
         System.err.println("NW_allowed = "+ NW_allowed);
         System.err.println("N_allowed = "+ N_allowed);
         System.err.println("NE_allowed = "+ NE_allowed);
         System.err.println("");
         System.err.println("use_Threshold = "+ use_Threshold);
         System.err.println("expandThreshold = "+ expandThreshold);
         System.err.println("reverseAlignDir = "+ reverseAlignDir);


         hide();
         dispose();
      }
```

```
      }

   }



//////////////////////////////////////////
//
// Main-Methode
//
   /** winziger Test: Lese Matrix aus Datei args[0] ein und aligne mal*/
   public static void main (String[] args){
      Matrix m = new Matrix(args[0]);
      AStarMatrixAligner asma = new AStarMatrixAligner(m);

      Pfad p = asma.aStarSearch(0,0,m.holeBreite()-1,m.holeHoehe()-1);
      System.out.println(""+p);

   }


}
```

M.5.18 de.denkselbst.matrixAligner.BestCellMatrixAligner.java

```
package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;


/**

   Diese Klasse löst das folgende Problem:
        Im de-news Korpus tritt das Phänomen auf, dass die Reihenfolge der Nachrichtenblöcke
   in der englischen Datei nicht mit der deutschen Datei übereinstimmt, obwohl zu jedem Block
   eine Übersetzung vorhanden ist.

   Idee aus dem Projekt / Arno: Man nehme aus beiden Dateien nur die Überschriften und aligne diese
   mit einem simplen Verfahren, dass Überkreuzungen zulässt. Auf diese Weise wird ein Mapping
   von englischen zu deutschen Überschriften erzeugt. Anhand dieses Mappings lassen sich die Dateien
   in dieselbe Reihenfolge bringen.

   Dieser MatrixAligner:
   a) sucht die beste Matrixzelle,
   b) trägt ihre Koordinaten als Mapping ein,
   c) zerstört die gesamte Spalte und Zeile, in der die Zelle liegt
   und d) wiederholt diese Schritte, bis zu jeder Koordinate der kürzeren Matrixdimension eine
   entsprechende Koordinate gefunden wurde, also
   zu jeder Zeile (Spalte) eine Spalte (Zeile) zugeordnet wurde.

   Auf diese Weise wird ein 1:1 Mapping von X zu Y (also englischer zu deutscher Überschrift) erzeugt.
   Momentan wird das Ergebnis als Pfad geliefert. Das Ergebnis ist korrekt, führt aber zu einer
   unschönen Darstellung in Mavis, da die Pfadkoordinaten nicht in geeigneter Reihenfolge im Pfad
   stehen, um in einem Linienzug gezeichnet zu werden.
   Konvertiert man den Pfad in eine Kokszuordnung, ist die Darstellung ok, die Reihenfolge jedoch
   unverändert.

   Ist die Matrix nicht quadratisch, gehen einige Spalten(Zeilen) leer aus, da auf einer 1:1
```

```
   Zuordnung bestanden wird.
  */


public class BestCellMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{


   /** Konstruktor */
   public BestCellMatrixAligner(Matrix matrix){
      super(matrix);
      this.alignerIdentString = "BestCell-1:1-MatrixAligner";     // Beschreibung setzen
   }


   /** Alignment-Methode */
   public Pfad align(){

      // Gib Statusinformation aus. (Wird an den Observer geschickt, der möglicherweise für eine Ausgabe sorg
      informObserver("Beginne 1:1 bestCell - Alignment", true);

      // Ergebnispfad
      Pfad p = new Pfad();    // Hier muss eigentlich eine KoksZuordnung hin!
                              // Das ist zu ändern!!!!

      // Höhe und Breite der Matrix besorgen
      int x = matrix.holeBreite();
      int y = matrix.holeHoehe();

      // lokale Kopie der Matrix erstellen, die bedenkenlos verändert werden kann
      Matrix lokal = new Matrix(matrix.kopieDerDaten());

      // Wert und Koordinaten der besten Zelle
      double bestValue;
      int bestX;
      int bestY;

      // Welche Dimension der Matrix ist kleiner?
      int mindim;

      boolean useX;
      if(x < y){ useX = true; mindim = x; }
      else{ useX = false; mindim = y; }

      // merken, ob Zeile schon für eine Zuordnung "benutzt" wurde
      boolean[] xBelegt = new boolean[mindim];                          // implizit mit false initialisiert

      // Schleifenende-Flagge
      boolean fertig = false;

      // solange wir noch nicht fertig sind
      while(!fertig){

         // Teste, ob cancel-Nachricht vom observer kam.
         if(cancelled) {
            informObserver("Alignment abgebrochen.",false);
            informObserverAboutFinishedAlignment(new Pfad(), false);
            return new Pfad();
         }

         // Gib Statusinformation aus.
         informObserver("Suche beste Zelle.", true);
```

```
        // besorge Koordinaten der besten Zelle
        bestX = lokal.getMinX();
        bestY = lokal.getMinY();

        // Nimm gefundene Koordinate in Lösungspfad auf
        p.haengeAn(bestX, bestY);

        // Gib Statusinformation aus.
        informObserver("Habe beste Zelle gefunden ("+bestX+", "+bestY+").", true);

        // merke, dass diese Spalte versorgt wurde
        if(useX) xBelegt[bestX] = true;
        else xBelegt[bestY] = true;

        // Gib Statusinformation aus.
        informObserver("Lösche zugehörige Zeile und Spalte.", true);

        // setze zur besten Zelle gehörige Zeile und Spalte auf unerreichbar schlechte Werte
        lokal.fillRow(bestY,666.6); // ein- / ausschaltbar machen!
        lokal.fillColumn(bestX, 666.6);

        // Gib Statusinformation aus.
        informObserver("Prüfe, ob fertig.", true);

        // Prüfen, ob alle Spalten versorgt sind
        boolean test = true;
        for(int i = 0; i < xBelegt.length; i++){
           test = test & xBelegt[i];                // Wenn eine Spalte nicht versorgt ist (=false), dann wird
        }
        fertig = test;
     }

     // Gib Statusinformation aus und liefere den Pfad.
     informObserver("BestCell1:1-Alignment abgeschlossen", true);
     informObserverAboutFinishedAlignment(p, true);
     return p;
  }


  /** Die run-Methode ruft lediglich align() auf. Ergebnisse werden über das Observer-Modell
      an den Interessenten geliefert. (Daher wird der Rückgabewert der Methode an dieser
      Stelle ignoriert.)
       */
  public void run(){
     align();
  }


  /** winziger Test: Lese Matrix aus Datei args[0] ein und gebe sie auf System.out aus. */
  public static void main (String[] args){
     Matrix m = new Matrix(args[0]);
     MatrixAligner aligner = new BestCellMatrixAligner(m);
     System.out.println("Alignment: "+aligner.align());
  }

}
```

M.5.19 de.denkselbst.matrixAligner.LMEDMatrixAligner.java

```
package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

import java.io.*;
import java.util.StringTokenizer;
import java.util.Iterator;

/**
  * Dieser Aligner basiert auf dem Levenshtein-Minimum-Edit-Distance Algorithmus. Er bietet eine bessere Lauf
  * als der AStarMatrixAligner, aber liefert auch schlechtere Ergebnisse.
  * Er entstand recht früh und in Zusammenarbeit mit Philip Reuter.
  */
public class LMEDMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

   double[][] leven;    // Levenshteinmatrix
   double[][] werte;    // Abstandsmatrix
   int levenX, levenY;     // Dimensionen

   /** Konstruktor */
   public LMEDMatrixAligner(Matrix matrix){
      super(matrix);
      this.alignerIdentString = "LMEDMatrixAligner";     // Beschreibung setzen
      levenX = matrix.holeBreite();
      levenY = matrix.holeHoehe();
      leven = new double[levenX][levenY];
      werte = new double[levenX][levenY];
      initWerte();
      initLeven();
   }

   private void initWerte(){
      double min = cost(0,0);
      double max = cost(0,0);
      double eps = 0.1;
      // Werte einlesen, dabei max und min berechnen
      for(int i=0; i < levenX; i++){
         for(int j=0; j < levenY; j++){
            werte[i][j] = cost(i,j);
            if(werte[i][j] < min){
               min = werte[i][j];
            }
            if(werte[i][j] > max){
               max = werte[i][j];
            }
         }
      }
      // Werte spreizen: (((x-min)*(1-eps))/(max-min))+eps
      for(int i=0; i < levenX; i++){
         for(int j=0; j < levenY; j++){
            werte[i][j] = (((werte[i][j]-min)*(1-eps))/(max-min))+eps;
         }
      }
   }

   private void initLeven(){
      // Extra-Spalte berechnen
      leven[0][0] = werte[0][0];
```

```
   for(int j=1; j < levenY; j++){
      leven[0][j] = leven[0][j-1] + werte[0][j];
   }
   // Extrazeile und Rest berechnen
   for(int i=1; i < levenX; i++){
      leven[i][0] = leven[i-1][0] + werte[i][0];

      for(int j=1; j < levenY; j++){
         leven[i][j] = minimalerVorgaenger(i,j) + werte[i][j];
      }
   }
}


private double cost (int x, int y){
   return matrix.holeDatumVonStelle(x,y);
}


// die geringsten Kosten zu x1,y1
private double minimalerVorgaenger(int x, int y){
   double diagonal = leven[x-1][y-1];
   double horizontal = leven[x-1][y];
   double vertikal = leven[x][y-1];
   return min(min(diagonal, horizontal), vertikal);
}

private double min(double c1, double c2){
   if (c1 <= c2) return c1;
   return c2;
}

public double get(int xx, int yy){
   return matrix.holeDatumVonStelle(xx,yy);
}

public int getXDim(){
   return matrix.holeBreite();
}

public int getYDim(){
   return matrix.holeHoehe();
}

/** Align-Methode */
public Pfad align(){

   finished = false;
   started = true;
   cancelled = false;
   int zzz = 0;         // Alive-Counter

   informObserver("LMED sprintet los.",true);   // Convenience-Methode aus der Oberklasse

   Pfad p = new Pfad();

   int i = matrix.holeBreite()-1;
   int j = matrix.holeHoehe()-1;

   p.haengeVorneAn(i,j);
```

```
      while( i>0 && j>0){

         zzz++;
         informObserver("LMED denkt angestrengt nach... "+zzz,true);

         double diagonal   = leven[i-1][j-1];
         double horizontal = leven[i-1][j];
         double vertikal   = leven[i][j-1];
         //double minV = min(min(diagonal,horizontal),vertikal);
         //double minV = minimalerVorgaenger(i,j);

         if(leven[i][j] - werte[i][j] == diagonal){
            i--;
            j--;
         }
         else if(leven[i][j] - werte[i][j] == horizontal){
            i--;
         }else {
            j--;
         }
         p.haengeVorneAn(i,j);

      }

      while(i > 0){
         p.haengeVorneAn(i-1,0);
         i--;

         zzz++;
         informObserver("LMED denkt angestrengt nach... "+zzz,true);
         if(cancelled) {
            informObserver("LMED Alignment abgebrochen.",false);
            informObserverAboutFinishedAlignment(new Pfad(), false);
         }


      }

      while(j > 0){
         p.haengeVorneAn(0,j-1);
         j--;

         zzz++;
         informObserver("LMED denkt angestrengt nach... "+zzz,true);
         if(cancelled) {
            informObserver("LMED Alignment abgebrochen.",false);
            informObserverAboutFinishedAlignment(new Pfad(), false);
         }

      }

      informObserver("LMED Alignment abgeschlossen.",true);
      informObserverAboutFinishedAlignment(p, true);
      finished = true;
      started = false;
      //System.out.println(p);
      return p;
   }

}
```

M.5.20 de.denkselbst.matrixAligner.BFSMatrixAligner.java

```
package de.denkselbst.matrixAligner;


import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.Comparator;
import java.util.Arrays;


import com.bluemarsh.adt.BinomialHeap;


/**
  * Dieser MatrixAligner verwendet eine Best-First-Suchstrategie. Die hier vorliegende Implementierung
  * diente vor allem dem Herumexperimentieren mit Strafen und eignet sich nur für kleine Matrizen. Der AStarMatr
  * ist schneller und benötigt weniger Speicher, liefert aber im Standartfall das gleiche Ergebnis.
  */
public class BFSMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

   private int x,y;
   private float steigung;

   // Fürs bfs-alignen: maximal ist maxHoriz:maxVert als Zuordnung in einem Block möglich!
   // Wege, die ein längeres vertikale, horizontales, rechts-hoch diagonales oder linksrunterdiagonales
   // Stück enthalten, als die folgenden Schwellen angeben, werden gelöscht
   private int maxHoriz = 2;  // 2
   private int maxVert = 2;    // 2

   private int maxHoRe = 2;   // 2 // rechts hoch
   private int maxLiRu = 2;   // 2 // links runter
   private boolean strangeDirsAllowed = false; //false (zu teuer!)// die beiden obigen Moves erlaubt?

   // Wie weit darf der Pfad von der Diagonalen entfernt sein?
   private int maxDistDiagX = 2; //3
   private int maxDistDiagY = 2; //3

   // Schwelle, die das Distanzmass unterschreiten muss, wenn Bewegung nach
   // hoch rechts (hore) oder liru (links runter)
   private double horeSchwelle = 0.3;
   private double liruSchwelle = 0.3;

   // Strafen für nichtdiagonale Bewegung
   private boolean bestrafeNichtdiagonaleBewegung = false; //false;
   private double rechtsStrafe = 0.1;
   private double runterStrafe = 0.1;

   // Wieviel kostet ein Schritt (Kante)
   private double kantenkosten = 0.2;
   private boolean benutzeKantenkosten = false; //true;
   // Gesamtkosten = Knotenkosten + Kantenkosten?




   public BFSMatrixAligner(Matrix matrix){
```

```
      super(matrix);
      this.alignerIdentString = "BFSMatrixAligner";      // Beschreibung setzen
      x = this.matrix.holeBreite();
      y = this.matrix.holeHoehe();

   /*
      // nur für TestZwecke
      if (x > 16) x = 16;
      if (y > 12) y = 12;
   */

      float xx = (float) x;
      float yy = (float) y;
      steigung = yy / xx;


   }


   // Kosten eines Knoten liefern
   public double kostenVonKnoten(Knoten k){
      return matrix.holeDatumVonStelle(k.xpos,k.ypos);
   }

   // Nachfolger eines Knoten liefern
   public ArrayList nachfolger(Knoten k){
      ArrayList n = new ArrayList();
      int x = k.xpos;
      int y = k.ypos;
      // Wo liegen die entsprechenden Diagonalen?
      int diaX = (int) (y / steigung);
      int diaY = (int) (x * steigung);
      int distX = 0;
      int distY = 0;

      if(x>0) distX = Math.abs(x - diaX);
      if(y>0)  distY = Math.abs(y - diaY);


      // Prüfen, ob zu weit von der Diagonalen entfern
      if( steigung < 1.0){
         if(distX > maxDistDiagX){
         //System.out.println("x zu weit weg");
            return new ArrayList();
         }
      }else{
         if(distY > maxDistDiagY){
         //System.out.println("y zu weit weg");
            return new ArrayList();
         }
      }

      /*
      if (!(distY <= maxDistDiagY && distX <= maxDistDiagX)){
      //System.out.println("Zu weit weg! Steigung:"+steigung+"   (maxx:"+maxDistDiagX+" x:"+distX+") ; maxy:"
         return new ArrayList();
      }
      */
```

```
   // schräg rechts runter
   if(x+1 < matrix.holeBreite() && y+1 < matrix.holeHoehe()){
      n.add(new Knoten(x+1, y+1));
   }

   // Alles, was nicht diagonal läuft, wird leicht bestraft
   // Alternative: Pauschalstrafe für alle = Luftlinie zum Zielpunkt


   // rechts
   if(x+1 < matrix.holeBreite()){
      Knoten knot=new Knoten(x+1, y);
      if(bestrafeNichtdiagonaleBewegung){knot.gibStrafe(rechtsStrafe);}
      n.add(knot);
   }

   // runter
   if(y+1 < matrix.holeHoehe()){
      Knoten knot=new Knoten(x, y+1);
      if(bestrafeNichtdiagonaleBewegung){knot.gibStrafe(runterStrafe);}
      n.add(knot);
   }

   // Optional: Bewegung in "seltsame" Richtungen
   if(strangeDirsAllowed){
      // schräg rechts hoch (evtl löschen!)
      if(x+1 < matrix.holeBreite() && y-1 >= 0 && y-1 < matrix.holeHoehe() && matrix.holeDatumVonStelle(x+1,y
         n.add(new Knoten(x+1, y-1));
      }

      // schräg links runter (evtl. löschen)
      if(x-1 >= 0 && x-1 < matrix.holeBreite() && y+1 > 0 && y+1 < matrix.holeHoehe() && matrix.holeDatumVonS
         n.add(new Knoten(x-1, y+1));
      }
   // endif strangedirs:
   }


   // Mal sehen, aus Spaß diese(n) Nachfolger mitanhängen...
   //if(y-1 > 0)  n.add(new Knoten(x, y-1));
   //if(x-1 > 0) n.add(new Knoten(x-1, y));

   //Dann aber auch Zyklencheck nötig! Haben wir...
   //*/

   return n;
}


// innere Klasse Knoten
public class Knoten{
   int xpos;
   int ypos;
   double kosten;

   public Knoten(int x, int y){
      xpos = x;
      ypos = y;
      kosten = kostenVonKnoten(this);
   }
```

```
    public double kosten(){
       return kosten;
    }

    public void gibStrafe(double strafe){
       kosten += strafe;
    }
 }

 public class Weg{
    ArrayList weg;
    double kosten;

    int vMov; //vMovementSinceNumberOfSquares;
    int hMov; //hMovementSinceNumberOfSquares;
    int hore;   // rechts hoch count
    int liru;   // links runter count
    int lastX;  // To track Movement
    int lastY;  // ""

    public Weg(){
       weg = new ArrayList();
       kosten = 0;
       vMov=0;  // Noch nicht bewegt.
       hMov=0;  // Noch nicht bewegt.
       lastX = lastY = 0; // ""
    }

    public void haengeKnotenAn(Knoten k){
             // Laufrichtung ermitteln
       int x = k.xpos;
       int y = k.ypos;

       if(x-lastX != 0 && y-lastY == 0){         // Horizontalbewegung (nur X-Achse)
          hMov++;
          vMov=0;
          hore=0;
          liru=0;

          //System.out.println("hMov++    "+hMov);
       }
       else if(y-lastY != 0 && x-lastX == 0){    // Vertikalbewegung (nur Y-Achse)
          vMov++;
          hMov=0;
          hore=0;
          liru=0;
          //System.out.println("hMov++    "+vMov);
       }
       else if(x-lastX>0  &&  y-lastY<0){        // rechts hoch
          vMov=0;                                // nur horiz. u. nur vert. Bewegung also seit 0 Knoten
          hMov=0;
          hore++;
          liru=0;
       }
       else if(x-lastX<0 && y-lastY>0){          // links runter
          vMov=0;                                // nur horiz. u. nur vert. Bewegung also seit 0 Knoten
          hMov=0;
          hore=0;
          liru++;
       }
       else{                                     // Bewegung auf beiden Achsen
```

```
      vMov=0;                                  // nur horiz. u. nur vert. Bewegung also seit 0 Knoten
      hMov=0;
      hore=0;
      liru=0;
   }
   lastX = x;
   lastY = y;

   weg.add(k);
   kosten += k.kosten();
   //System.out.println("haenge an, Kosten gesamt: "+kosten);

}

public boolean bistDuZuLangeHorizontalOderVertikalGelaufen(){
   //if(hMov>=maxHoriz || vMov>=maxVert) System.out.println("maximum vert/horiz violated.");
   //System.out.println("weg hMov:"+hMov+" vMov:"+vMov+" violatedLimits:"+!(hMov<maxHoriz && vMov<maxVert)
   return (!(hMov<maxHoriz && vMov<maxVert && hore < maxHoRe && liru < maxLiRu));
}

public Iterator iterator(){
   return weg.iterator();
}

public double kosten(){
/* double sum = 0;
   for (int i = 0; i < weg.size(); i++){
      sum += ((Knoten)weg.get(i)).kosten();  // Kosten der Knoten aufsummieren
      // Hier später noch die Kosten der Kante hinzuaddieren!!!
   }
   return sum;*/

   // Weit weg vom Ziel sein wird bestraft, allerdings unterschätzend
   double entfernungsPauschale = 0;

   int entfSchritte = 0;

   double kost=0;

   // Entfernungspauschale Strafe besorgen
   if(!weg.isEmpty()){
      Knoten k = (Knoten) weg.get(weg.size()-1);
      entfernungsPauschale =  matrix.wasSindVonHierDieUnterschaetztenKostenZurRechtenUnterenEcke(k.xpos, k
      entfSchritte = matrix.wievieleSchritteVonHierZurRechtenUnterenEcke(k.xpos, k.ypos);
   //System.out.println("Entfernungspauschale von ("+k.xpos+", "+k.ypos+") zum Ziel: "+entfernungsPauschal
   //System.out.println("Gesamtkosten: "+kosten + entfernungsPauschale);
   }

   kost = kosten + entfernungsPauschale;

   kost = kost * kost;

   // Wenn Kanten als Kosten auch mitgezählt werden sollen:
   if (benutzeKantenkosten){System.out.println("benutzeKantenkosten");
      kost += entfSchritte * kantenkosten;
      // kost += (weg.size()-1) * kantenkosten;
   }
   //System.out.println("Kosten: "+kost);
   return kost;
}
```

```
      public boolean enthaeltstDuDiesenKnoten(Knoten k){
         if (weg.contains(k)) System.out.println("potentieller Zyklus");
         return weg.contains(k);
      }

      public boolean endestDuInDiesemKnoten(Knoten k){
         return endestDuIn(k.xpos, k.ypos);
      }

      public boolean endestDuIn(int x, int y){
         if(weg.isEmpty()) return false;
         int l = weg.size() - 1;
         Knoten k = (Knoten)weg.get(l);
         if(x == k.xpos && y == k.ypos) return true;
         return false;
      }

      public ArrayList nachfolgeWege(){
         if(weg.isEmpty()) return new ArrayList();
         int l = weg.size() - 1;
         Knoten letzter = (Knoten)weg.get(l);
         ArrayList n = nachfolger(letzter);

         ArrayList nwege = new ArrayList();
         for(int i = 0; i < n.size(); i++){
            // Zyklencheck
            if(enthaeltstDuDiesenKnoten((Knoten)n.get(i))) continue;
            // Weg kopieren
            Weg neuerWeg = new Weg();
            for(int j = 0; j < weg.size(); j++){
               neuerWeg.haengeKnotenAn((Knoten)weg.get(j));
            }
            // Nachfolgeknoten anhängen
            neuerWeg.haengeKnotenAn((Knoten)n.get(i));
            //System.err.println("neuerWegKosten: "+neuerWeg.kosten()+" alter: "+kosten());
            // Prüfen, ob maximale horizontale oder vertikale Bewegung am Stück überschritten
            if(neuerWeg.bistDuZuLangeHorizontalOderVertikalGelaufen()){
            //System.out.println("foo, zu lange in x oder y gelaufen, weg damit!!");
               continue;
            }
            // Alles klar, Weg ist gültig, also in die Nachfolgerliste aufnehmen!
            nwege.add(neuerWeg);
         }
         //System.out.println("+"+nwege.size());
         return nwege;
      }

      // Billig Textausgabe
      public void print(){
         //System.out.println("...");
         for(int i = 0; i < weg.size(); i++){
            System.out.print("["+((Knoten)weg.get(i)).xpos+",");
            System.out.print(((Knoten)weg.get(i)).ypos+"]");
         }
         //System.out.println("...");
      }

   }

/** Innere Klasse: Agenda für die Breitensuche */
public static class Agenda{
```

```
   BinomialHeap heap;

   public Agenda(){
      heap = new BinomialHeap();
   }

   public boolean bistDuLeer(){
      return heap.empty();
   }

   public int size(){
      return heap.size();
   }

         // Einen Weg einfügen
      public void fuegeEinenWegEin(Weg w){
         add(w.kosten(),w);
      }

      // Liste von Wegen einfügen
      public void fuegeWegeEin(ArrayList wege){
         for(int i = 0; i < wege.size(); i++){
            fuegeEinenWegEin((Weg)(wege.get(i)));
         }
      }


   /** Objekt einfügen */
   public void add(double val, Object o){
      heap.insert(o,val);
   }


   public Weg billigster(){
      return (Weg)  heap.removeMin().data;
   }



   public void behalteNurSoviele(int anzahl){

      BinomialHeap h2 = new BinomialHeap();
      BinomialHeap.Node n;

      for(int i=0; i< anzahl; i++){
         n=heap.removeMin();
         h2.insert(  n.data, ((Weg)n.data).kosten());

      }
      heap = h2;

   }

}


   // Mittels BFS Pfad von 0,0 nach x,y suchen
   public Weg bfsalign(){

      cancelled=false;
```

```
      // Agenda initialisieren
      Weg start = new Weg();
      start.haengeKnotenAn(new Knoten(0,0));
      Knoten ziel = new Knoten(matrix.holeBreite()-1, matrix.holeHoehe()-1);
      Agenda ag = new Agenda();
      ag.fuegeEinenWegEin(start);


      for(;;){
      //System.out.println("Es sind "+ag.size()+" Wege in der Agenda.");

         // observer Statusbericht geben
         if(ag.size()==0)  informObserver("Die Agenda ist leer.", false);
         else informObserverZZZ("Es sind "+ag.size()+" Wege in der BFS-Agenda.",true);

         if(cancelled) {
            informObserver("Alignment abgebrochen.",false);
            informObserverAboutFinishedAlignment(new Pfad(), false);
            observer=null; // Observer vergessen
         }

         // Agenda leer? -> misserfolg.
         if(ag.bistDuLeer()){
            return new Weg();
         }


         // cancelled Flagge gesetzt?
         if(cancelled){
         //System.err.println("abgebrochen!");
            informObserverAboutFinishedAlignment(new Pfad(), false);
            observer=null; // Observer vergessen
         ///// Irgendwo muss noch geregelt werden, dass sich diese Aligner bei misserfolg automatisch
         ///// von der Matrix abmeldet!!!
            return new Weg();
         }


         // ersten Weg rausholen
         Weg billig = ag.billigster();

         // am Ziel? -> Weg liefern
         if(billig.endestDuInDiesemKnoten(ziel)){
            System.err.println("Ziel erreicht: ");
//          System.err.println("Der Wehk kostät: "+billig.kosten()+"! violation:"+billig.bistDuZuLangeHorizon
            billig.print();
            // Observer benachrichtigen.
            informObserver("Alignment abgeschlossen. "+ag.size()+" Wege.",true);
            // Observer den Weg als Pfad geben


            Iterator it = billig.iterator();
            Pfad pf = new Pfad();
            Knoten kn=null;
            while(it.hasNext()){                      // Weg in Pfad umwandeln
               kn = (Knoten)it.next();
               pf.haengeAn(kn.xpos,kn.ypos);
            }
            informObserverAboutFinishedAlignment(pf, true); // und beim Observer abliefern...
```

```
            observer=null; // Observer vergessen
                  //von der Matrix abmelden?!?!
            return billig;
         }
         else{ // Nein, expandiere und fuege Nachfolger ein
            ag.fuegeWegeEin(billig.nachfolgeWege());
         }
      }
   }

   /** Alignment als Pfad liefern */
   public Pfad align(){

      finished = false;
      cancelled = false;
      started = true;

      Iterator it = bfsalign().iterator();      // bfsalign liefert Weg
      Pfad p = new Pfad();
      Knoten k=null;

      while(it.hasNext()){                      // Weg in Pfad umwandeln
         k = (Knoten)it.next();
         p.haengeAn(k.xpos,k.ypos);
      }

      finished = true;
      started = false;
      return p;                                 // Pfad liefern
   }



   /** run-Methode ruft nur bfsalign() auf... */
   public void run(){
      bfsalign();
   }


   /** Test / Demo: Lese Matrix aus Datei args[0] ein und gebe Alignment auf System.out aus. */
   public static void main (String[] args){
      Matrix m = new Matrix(args[0]);
      MatrixAligner bfs = new BFSMatrixAligner(m);
      System.out.println("Alignment: "+bfs.align());
   }

}
```

M.5.21 de.denkselbst.kommandozeile.Align.java

```
package de.denkselbst.kommandozeile;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrixAligner.AStarMatrixAligner;

import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
```

```
/**
  * Die Main-Methode dieser Klasse liest Matrizen von STDIN und
  * schreibt Alignments nach STDOUT.
  * Die wichtigsten Schritte sind es, den Eingabestrom zu besorgen,
  * ein Matrixobjekt und ein damit verbundenes MatrixAlignerobjekt zu
  * erstellen, die Matrix aus STDIN zu füllen, zu normalisieren,
  * den MatrixAligner das Alignment (als Pfad) finden lassen, den
  * Pfad in eine KoksZuordnung umzuwandeln und das
  * Ergebnis auf STDOUT auszugeben.
  *
  * Da es offensichtliche Performanzvorteile gibt, wenn nicht für
  * jede Matrix einmal die JAVA-VM gestartet werden muss, können
  * align mehrere Aufträge pro Aufruf erteilt werden. Dazu muss der
  * erste Parameter ein Integer-Wert sein, der die Anzahl der Aufträge
  * enthält. Um den obigen Code-Ausschnitt herum existiert eine Schleife.
  * Wird align ohne Angabe der Auftragsanzahl aufgerufen, wird diese
  * Schleife nur einmal durchlaufen.
*/

public class Align{
   public static void main(String args[]) throws IOException{
      // Stdin-Eingabestrom besorgen.
      BufferedReader puff = new BufferedReader(new InputStreamReader(System.in));

      /**
        Anmerkung: die Methode matrix.setzeDatenAusStdin() kann in diesem Szenario nicht
        verwendet werden, da der BufferedReader des ersten Matrix-Objektes seinen Puffer
        mit Daten aus Stdin auffüllt, die schon zur zweiten Matrix gehören. Diese gepufferten
        Daten gehen verloren, sobald das erste Matrix-Objekt zerstört wird. Damit wird der
        Ansatz unbrauchbar. Um dies zu verhindern, wird der BufferedReader von Main gehalten,
        und die Matrix-Objekte lesen nur so viel, wie sie benötigen.
      */

      int anzahl = 0;               // Anzahl zu lesender Matrizen

      if(args.length == 0){
         anzahl = 1;          // eine Matrix ist zu lesen
      }
      else if(args.length == 1){
         anzahl = Integer.parseInt(args[0]); // 1. Parameter enthält Matrixanzahl
      }else{
         System.err.println("Kommandozeilenaligner liest Matrix von stdin, schreibt alignment nach stdout");
         System.exit(1);
      }

   // Aufträge abarbeiten
   while(anzahl-->0){

      // Matrixobjekt anlegen
      Matrix matrix = new Matrix();

      // Alignerobjekt anlegen
      AStarMatrixAligner aligner = new AStarMatrixAligner(matrix);

      // besorge Matrix aus Stdin, Aligner bekommt die Matrixänderung
      // automatisch von der Matrix mitgeteilt
      matrix.setzeDatenAusGepuffertemLeser(puff);

      // Normalisiere Matrix
      matrix.normalisiereDich();
```

```
      // aligne
      Pfad p = aligner.align();

      // schreibe Ergebnis
      System.out.print(p.toWagnerString(matrix.holeQuelle(),matrix.holeInfo()));

      }
   }
}
```

M.5.22 de.denkselbst.kommandozeile.HeadingMapper4Arno.java

```
package de.denkselbst.kommandozeile;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrixAligner.BestCellMatrixAligner;
import de.denkselbst.matrixAligner.MatrixAligner;

import java.io.IOException;

/**
  * Die Main-Methode dieser Klasse liest Matrizen von Stdin
  * und schreibt 1:1 Zuordnungen im Wagner-Format nach Stdout.
  * Sonstige Meldungen werden auf Stderr ausgegeben.
  * Idee und Algorithmus sind in der Dokumentation der Klasse
  * de.denkselbst.matrixAligner.BestCellMatrixAligner beschrieben.
  */
public class HeadingMapper4Arno{
   public static void main(String args[]) throws IOException{

      if(args.length > 0){
         System.err.println("HeadingMapper4Arno liest "+
                  "Matrix von stdin, schreibt 1:1-Mapping "+
                  "nach stdout");
         System.exit(1);
      }

      // Matrixobjekt anlegen
      Matrix matrix = new Matrix();

      // Alignerobjekt anlegen
      MatrixAligner aligner = new BestCellMatrixAligner(matrix);

      // besorge Matrix aus Stdin, Aligner bekommt die Matrixänderung
      // automatisch von der Matrix mitgeteilt (Observer-Modell)
      matrix.setzeDatenAusStdin();

      // Normalisiere Matrix
      matrix.normalisiereDich();

      // aligne
      Pfad p = aligner.align();

      // schreibe Ergebnis
      System.out.println(p.toWagnerString(matrix.holeQuelle(),matrix.holeInfo()));
   }
}
```

M.5.23 de/denkselbst/kommandozeile/manifest

```
Main-Class: de.denkselbst.kommandozeile.Align
```

M.5.24 de/denkselbst/kommandozeile/manifest2

```
Main-Class: de.denkselbst.kommandozeile.HeadingMapper4Arno
```

M.5.25 de.denkselbst.beobachtung.Beobachter.java

```
package de.denkselbst.beobachtung;

/**
 *  Wenn ein beobachtetes Objekt eine Änderung zu melden hat, ruft es die in diesem
 *  Interface deklarierte Methode bei seinen Beobachtern auf.
 */

public interface Beobachter{
   /**
     * Diese Methode wird von dem beobachteten Objekt ausgelöst, wenn
     * eine Änderung erfolgt ist. Das Objekt liefert sich selbst und
     * eine Nachricht (String) beim Aufruf mit.
     */
   public void beobachtetesObjektHatSichGeregt(Object o, String nachricht);
}
```

M.5.26 de.denkselbst.beobachtung.Beobachtbar.java

```
package de.denkselbst.beobachtung;

/**
  * Dieses Interface legt die Methoden fest, die ein Objekt implementieren muss, damit
  * es von anderen Objekten beobachtet werden kann.
  */

public interface Beobachtbar{

   /**
     * einen neuen Beobachter in die Liste der im Falle einer Änderung zu benachrichtigen
     * Objekte aufnehmen
     * @param b das Objekt, das Interesse an Änderungen hat
     */
   public void beobachterAnmelden (Beobachter b);

   /**
     * einen Beobachter aus der Liste der zu benachrichtigen
     * Objekte entfernen.
     * @param b das Objekt, das das Interesse an den Änderungen verloren hat
     */
   public void beobachterAbmelden (Beobachter b);
}
```

M.5.27 de.denkselbst.mavis.Mavis.java

```
package de.denkselbst.mavis;

import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrix.KoksZuordnung;
import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.MatrixView;
import de.denkselbst.matrix.MatrixHistogramView;
```

```
import de.denkselbst.matrixAligner.MatrixAligner;
import de.denkselbst.matrixAligner.BFSMatrixAligner;

import java.io.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.JOptionPane;
import java.awt.event.ItemListener;
import java.awt.event.*;
import javax.swing.event.*;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JCheckBoxMenuItem;

/**
  * Diese Klasse setzt die grafische Oberfläche von Mavis zusammen und kümmert sich um Teile des Event-Handlings
  * Der Code ist immer mal wieder auf die Schnelle geändert worden. Er ist nicht besonders schön und auch mit de
  * heißen Nadel gestrickt. Aber das muss so sein, schließlich handelt es sich um einen Prototypen.
  */

public class Mavis extends JFrame implements ActionListener, ItemListener{

   /** Speichert aktuellen Dateinamen der Matrix*/
   private String filename;
   private MatrixView mv;
   private JScrollPane sp;
   private Matrix matrix;

   /** Referenz auf Histogramm-Fenster*/
   private JFrame hist;
   private MatrixHistogramView histView;

   /** enthält aktuellen Dateinamen..*/
   private JLabel la;

   //Create a file chooser
   private final JFileChooser fc = new JFileChooser();


   public Mavis(Matrix mat, MatrixView mv, String title){
      hist = null;
      histView = null;

      filename = title;
      this.mv = mv;
      this.matrix = mat;

      sp = new JScrollPane(mv, JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED, JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEE
      sp.setMinimumSize(new Dimension(320,200));   // Mindestgröße von 320 * 200 einfordern

      // MatrixView sagen, dass sie in dieser Scrollpane steckt! (für MatrixView.XYInfo)
      //mv.setzeScrollPane(sp);

      setTitle(title+" - Mavis");
      getContentPane().add(sp, BorderLayout.CENTER);

   // Menüs erstellen
      JRadioButtonMenuItem rbMenuItem;
      JCheckBoxMenuItem cbMenuItem;
         JMenuBar menuBar;
```

```
      JMenu menu, submenu;
   JMenuItem menuItem;
menuBar = new JMenuBar();
setJMenuBar(menuBar);

menu = new JMenu("Matrix");
menuBar.add(menu);
menuItem = new JMenuItem("öffnen");
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem("zurücksetzen");
menuItem.addActionListener(this);
menu.add(menuItem);

menu.addSeparator();

menuItem = new JMenuItem("Normalisieren");
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem("RangfolgenClustering");
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem("stutzen");
menuItem.addActionListener(this);
menu.add(menuItem);

menu.addSeparator();

menuItem = new JMenuItem("Histogramm-Fenster");
menuItem.addActionListener(this);
menu.add(menuItem);


menu = new JMenu("Alignment");
menuBar.add(menu);
menuItem = new JMenuItem("A*");
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("Best First Search");
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("Levenshtein Minimum Edit Distance");
menuItem.addActionListener(this);
menu.add(menuItem);

menu.addSeparator();

menuItem = new JMenuItem("Best Cell 1:1");
menuItem.addActionListener(this);
menu.add(menuItem);

menu.addSeparator();


menuItem = new JMenuItem("selektiertes Alignment speichern");
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("Alignment laden und anzeigen");
```

```
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("selektiertes Alignment textuell anzeigen");
menuItem.addActionListener(this);
menu.add(menuItem);

menu.addSeparator();

menuItem = new JMenuItem("selektiertes Alignment löschen");
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("Alle Alignments löschen");
menuItem.addActionListener(this);
menu.add(menuItem);

menu = new JMenu("Ansicht");
menuBar.add(menu);
submenu = new JMenu("Blockgröße");
menu.add(submenu);
menuItem = new JMenuItem("30");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("20");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("18");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("16");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("14");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("12");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("10");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("8");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("6");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("4");
submenu.add(menuItem);
menuItem.addActionListener(this);

menu.add(submenu);
```

```
      cbMenuItem = new JCheckBoxMenuItem("Diagonale anzeigen");
      cbMenuItem.addItemListener(this);

      menu.add(cbMenuItem);


      la = new JLabel(""+title);
      getContentPane().add(la, BorderLayout.SOUTH);

      JButton rl = new JButton("reload");
      rl.addActionListener(this);
      //getContentPane().add(rl, BorderLayout.SOUTH);


      addWindowListener( new WindowAdapter(){
            public void windowClosing(WindowEvent we){
               setVisible(false);
               dispose();
               System.exit(0);
            }
         }

      );

      pack();
      setLocation(220,100);
      show();

   }


   public void actionPerformed(ActionEvent e){
      String cmd = e.getActionCommand();

      /** Hier muss unbedingt aufgeräumt werden if-else if-else if, bzw
          innere Klassen, ...
          (Sobald mal feststeht, was alles wie aussehen soll...)
           */

      System.err.println("Action: "+cmd);

      if (cmd.equals("zurücksetzen")){
         if(matrix!=null) matrix.setzeDatenAusDatei(filename);
      }

      if(cmd.equals("öffnen")){
         int returnVal = fc.showOpenDialog(this);
         if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            System.err.println("Opening: " + file.getName() + "." );


            if(matrix!=null) matrix.setzeDatenAusDatei(file.toString());
            else{
               matrix = new Matrix(file.toString());
               mv.setzeMatrix(matrix);
            }


            setTitle(file.toString()+" - Mavis");
            la.setText(file.toString());
```

```
      filename = file.toString();   // Dateinamen merken...
      if(hist != null) hist.setTitle("Histogramm - "+file.toString()+" - Mavis");
         // alte Alignmentpfade wegbürsten
         mv.loescheAllePfade();


   }else {
      System.err.println("Open command cancelled by user." );
   }
}

if(cmd.equals("RangfolgenClustering")){
   String inputValue = JOptionPane.showInputDialog(this, "Wie viele Klassen?");
   int val;
   if(inputValue == null) return;
   try{
      val = Integer.parseInt(inputValue);
   }catch(NumberFormatException nmfe){
      JOptionPane.showMessageDialog(this, "Konnte die Eingabe "+inputValue+" nicht als Zahl interpretieren
      val = 10;
   }
   val = Math.abs(val);
   matrix.rangfolgenClustering(val);
}

if(cmd.equals("Normalisieren")){
   matrix.normalisiereDich();
}




if(cmd.equals("stutzen")){
   StutzenDialog s = new StutzenDialog(this, matrix.holeBreite(), matrix.holeHoehe());
   if(s.cancelled()!=true){
      int xn = s.getXVal();
      int yn = s.getYVal();
      matrix.stutzenAuf(xn,yn);
      // alte Alignmentpfade wegbürsten
      mv.loescheAllePfade();
   }
   s.dispose();
}

if(cmd.equals("Alle Alignments löschen")){
   mv.loescheAllePfade();
}
if(cmd.equals("selektiertes Alignment löschen")){
   mv.entferneSelektiertenAlignmentPfad();
}

if(cmd.equals("selektiertes Alignment textuell anzeigen")){
   mv.zeigeSelektiertenAlignmentPfadAlsHTML();
}

if(cmd.equals("selektiertes Alignment speichern")){

   // Checken, ob was selektiert ist!!!
   if (!mv.istEtwasSelektiert()) return;

   JFileChooser fc = new JFileChooser();
```

```
         int returnVal = fc.showSaveDialog(this);

         if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            //this is where a real application would save the file.
            mv.speichereSelektiertenAlignmentPfad(file);

            System.err.println("Pfad gepeichert in "+file.toString());
         } else {
            System.err.println("Pfad Speichern abgebrochen...");
         }


      }



      if(cmd.equals("Alignment laden und anzeigen")){
         int returnVal = fc.showOpenDialog(this);
         if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            System.err.println("Opening: " + file.getName() + "." );

            KoksZuordnung al = new KoksZuordnung(file.toString());

            mv.fuegeKoksZuordnungEin(al,file.toString() );

         }else {
            System.err.println("Open command cancelled by user." );
         }
      }




// verschiedene Alignments abfahren
//
      if(cmd.equals("A*")){
         MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.AStarMatrixAligner");
         macv = null; // Ist das Ok? Wenn ja warum? Wenn nein, warum nicht? Wartet macv im Konstruktor, bis a
      }


      if(cmd.equals("Best First Search")){
         MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.BFSMatrixAligner");
         macv = null;
      }


      if(cmd.equals("Levenshtein Minimum Edit Distance")){
         MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.LMEDMatrixAligner");
         macv = null;
      }


      if(cmd.equals("Best Cell 1:1")){
         MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.BestCellMatrixAligner");
         macv = null;
      }
```

```
if(cmd.equals("30")){mv.setzeBlockGroesse(30,30);}
if(cmd.equals("20")){mv.setzeBlockGroesse(20,20);}

if(cmd.equals("18")){mv.setzeBlockGroesse(18,18);}
if(cmd.equals("16")){mv.setzeBlockGroesse(16,16);}
if(cmd.equals("14")){mv.setzeBlockGroesse(14,14);}
if(cmd.equals("12")){mv.setzeBlockGroesse(12,12);}

if(cmd.equals("10")){mv.setzeBlockGroesse(10,10);}
if(cmd.equals("8")){mv.setzeBlockGroesse(8,8);}
if(cmd.equals("6")){mv.setzeBlockGroesse(6,6);}
if(cmd.equals("4")){mv.setzeBlockGroesse(4,4);}

if(cmd.equals("Histogramm-Fenster")){
   //Ist schon eins offen?
   if(histView != null){

      // Aufhören, die Matrix zu observen!
      histView.abmelden();
      //
      // Fenster wegwerfen
      hist.dispose();
   }
   //nein, dann neu


   /* Histogramm-Fenster */
   // Bastele Histogramm-Fenster
   hist = new JFrame("Histogramm - "+filename+" - Mavis");
   histView = new MatrixHistogramView(matrix);
   hist.getContentPane().add( histView, BorderLayout.CENTER );
   hist.setSize(new Dimension(220,70));
   hist.setResizable(false);
   hist.setLocation(0,300);


   hist.addWindowListener( new WindowAdapter(){
         public void windowClosing(WindowEvent event){
            histView.abmelden();
            histView = null;
            hist.setVisible(false);
            hist.dispose();
         }
      }
   );

   // Radiobuttons einfügen
     JRadioButton sg02 = new JRadioButton("05");
     sg02.setActionCommand("0.2");
     JRadioButton sg01 = new JRadioButton("10");
     sg01.setActionCommand("0.1");
     JRadioButton sg005 = new JRadioButton("20");
     sg005.setActionCommand("0.05");
     sg005.setSelected(true);
     JRadioButton sg0025 = new JRadioButton("40");
     sg0025.setActionCommand("0.025");
     JRadioButton sg00125 = new JRadioButton("80");
     sg00125.setActionCommand("0.0125");
   // Group the radio buttons.
```

```
            ButtonGroup group = new ButtonGroup();
            group.add(sg02);
            group.add(sg01);
            group.add(sg005);
            group.add(sg0025);
            group.add(sg00125);

         // Register a listener for the radio buttons.
         /** Listens to the radio buttons. */
         class RadioListener implements ActionListener {
         public void actionPerformed(ActionEvent e) {

            histView.setzeSchrittgroesse(Double.parseDouble(e.getActionCommand()));

            }
         }

         RadioListener myListener = new RadioListener();
         sg02.addActionListener(myListener);
         sg01.addActionListener(myListener);
         sg005.addActionListener(myListener);
         sg0025.addActionListener(myListener);
         sg00125.addActionListener(myListener);

         // Put the radio buttons in a column in a panel
         JPanel radioPanel = new JPanel();
         radioPanel.setLayout(new GridLayout(0,1));
         radioPanel.add(sg02);
         radioPanel.add(sg01);
         radioPanel.add(sg005);
         radioPanel.add(sg0025);
         radioPanel.add(sg00125);
         //setLayout(new BorderLayout());
         hist.getContentPane().add(radioPanel, BorderLayout.WEST);

          hist.show();


      }

   }

   public void itemStateChanged(ItemEvent e){
// jaja, ist schlecht, da eine zweite Checkbox nicht von der ersten unterschieden werden kann...
System.err.println("ItemEvent: "+e.getStateChange());
      if(e.getStateChange()==ItemEvent.SELECTED){
         mv.zeigeDiagonale(true);
      }else{
         mv.zeigeDiagonale(false);
      }
   }


   /** Main-Methode bringt den Stein ins Rollen. */
   public static void main(String[] args) throws IOException{


      //System.err.println(""+System.getProperty("user.dir"));
      //System.err.println(""+System.getProperty("java.class.path"));
```

```
      String title = "[keine Datei geladen]" ;

      Matrix matrix = null;
      // Besorge Matrix aus Datei
      if(args.length > 0){
         matrix = new Matrix(args[0]);
         title = args[0];
      }else{
   //    matrix = new Matrix(10,10);
      }


      // Bastele MatrixView dazu
      //MatrixView mv = new MatrixView(matrix);
      MatrixView mv = new MatrixView();
      mv.setzeMatrix(matrix);

      // Bastele Mavis dazu
      Mavis m = new Mavis(matrix, mv, title);


   }




}
```

M.5.28 de.denkselbst.mavis.MACV.java

```
package de.denkselbst.mavis;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrix.MatrixView;
import de.denkselbst.matrixAligner.MatrixAligner;
import de.denkselbst.matrixAligner.AlignmentObserver;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Dimension;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;


import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
/**
  * Objekte dieser Klasse rufen einen MatrixAligner ins Leben und observieren ihn.
  * Die Nachrichten des Aligners werden in einem eigenen Fenster dargestellt,
  * das von dieser Klasse erzeugt wird.
  * Objekte dieser Klasse werden von Mavis.java ins Leben gerufen, wenn der
  * Benutzer ein Alignment veranlasst.
  */
public class MACV extends JFrame implements ActionListener, AlignmentObserver{
```

```
private Matrix m;
private MatrixAligner aligner;

private JLabel status;
private JButton cancelButton;
private JButton okButton;
private MatrixView mv;


/**
  * Konstruktor.
  * @param m Matrix, die die Daten zum Alignen liefert
  * @param mv MatrixView, die das Ergebnis bekommen soll
  * @param klasse Klasse, von der ein MatrixAligner instantiiert werden soll
  * (Z.B. de.denkselbst.matrixAligner.BFSMatrixAligner, de.denkselbst.matrixAligner.LMEDMatrixAligner,
  * de.denkselbst.matrixAligner.AStarMatrixAligner...)
  */
public MACV(Matrix m, MatrixView mv, String klasse){

   boolean ok = true;         // Flagge, ob nachfolgende Reflection-Aktionen geklappt haben und
                              // ein MatrixAligner instantiiert werden konnte

   // von MatrixAligner abstammendes Objekt der Klasse klasse anlegen
   // bzw. versuchen, es anzulegen
   try{
      // Wenn Matrix m == null, dann können wir uns das alles schenken...
      if(m == null){
         throw new Exception("Keine Matrix vorhanden! Laden Sie eine!");
      }

      // Klasse zum String klasse besorgen
      Class alignClass = Class.forName(klasse);

      // formale Parameter erzeugen
      Class parameter[] = new Class[1];
      parameter[0] = m.getClass();
      // passenden Konstruktor besorgen
      Constructor alignConstructor = alignClass.getConstructor(parameter);

      // Argumente für den Konstruktoraufruf erzeugen
      Object argumente[] = new Object[1];
      argumente[0] = m;
      // Konstruktor mit Argumenten aufrufen und damit MatrixAligner-Objekt erzeugen
      aligner = (MatrixAligner) alignConstructor.newInstance(argumente);

   }  // Mögliche Exceptions behandeln
   catch(ClassNotFoundException e){
      ok = false;
      System.err.println(e.getMessage());
      e.printStackTrace();
   }catch(NoSuchMethodException e){
      ok = false;
      System.err.println(e.getMessage());
      e.printStackTrace();
   }catch(InstantiationException e){
      ok = false;
      System.err.println(e.getMessage());
      e.printStackTrace();
   }catch(IllegalAccessException e){
      ok = false;
```

```
   System.err.println(e.getMessage());
   e.printStackTrace();
}catch(IllegalArgumentException e){
   ok = false;
   System.err.println(e.getMessage());
   e.printStackTrace();
}catch(InvocationTargetException e){
   ok = false;
   System.err.println(e.getMessage());
   e.printStackTrace();
}catch(Exception e){
   ok = false;
   System.err.println(e.getMessage());

}finally{
   if(!ok){
      System.err.println("Es gab bei dem Versuch ein MatrixAligner Objekt"+
                         " der Klasse "+klasse+" anzulegen eine Exception. (siehe oben)");
   }else{
      System.err.println("MatrixAligner der Klasse "+klasse+" angelegt.");
   }
}

this.mv = mv;                                        // MatrixView merken
this.m = m;                                          // Matrix merken

JLabel label = new JLabel();                         // Label anlegen
status = new JLabel();                               // Status-Label anlegen
cancelButton = new JButton("Abbrechen!");            // Abbrechen-Knopf anlegen
cancelButton.setEnabled(true);
cancelButton.addActionListener(this);
okButton = new JButton("OK");                        // Ok-Knopf anlegen
okButton.setEnabled(false);
okButton.addActionListener(this);

if(ok && (m != null)){                         // Aligner sowie Matrix vorhanden
   String temp = aligner.alignerInfo();
   setTitle(temp);                                   // Fenstertitel setzen
   label.setText("Hier arbeitet ein "+temp+".");     // Label setzen
   status.setText("Starte...");
}
else{                                                // kein Aligner vorhanden
   setTitle("Achtung!");                             // Fenstertitel setzen
   label.setText("Der Computer hat die Arbeit niedergelegt.");

   if(m==null) status.setText("Grund: Es ist noch keine Matrix geladen worden.");
   else status.setText("Grund: Das MatrixAligner-Objekt konnte nicht angelegt werden.");

   cancelButton.setText("Was solls!");
}

// GUI-Objekte in das Frame einhängen
getContentPane().add(label, BorderLayout.NORTH);
getContentPane().add(status, BorderLayout.CENTER);
JPanel btnPanel = new JPanel(new FlowLayout());
btnPanel.add(cancelButton);
btnPanel.add(okButton);
getContentPane().add(btnPanel, BorderLayout.SOUTH);

pack();                                              // Layout zusammenschieben
show();                                              // Fenster anzeigen
```

```
      // nur wenn alles OK ist, es also einen Aligner gibt, wird mit uns als observer alignt.
      if(ok){
         aligner.konfigurationsDialogAbfahren();         // Konfigurationsmethode anstossen
         aligner.align(this);                            // Alignen...
      }

   }


   /** ActionListener */
   public void actionPerformed(ActionEvent e){

      if(e.getSource() == okButton){
         if(aligner!=null)aligner.abmelden();      // Meldet den Aligner von der Matrix ab,
         aligner = null;                           // Frei zum garbage-collecten
         dispose();                                // damit keine Referenzen mehr auf ihn existieren und er
      }                                            // garbage-collected werden kann
      if(e.getSource() == cancelButton){
         if(aligner!=null){
            aligner.cancel();                      //    "
            aligner.abmelden();                    //    "
            aligner = null;                        // Frei zum garbage-collecten
         }
         dispose();
      }
   }


   /** AlignmentObserver */
   public void alignmentFertig(Pfad p, boolean erfolg){
      okButton.setEnabled(true);
      cancelButton.setEnabled(false);
      if(erfolg){
         mv.fuegeAlignmentPfadHinzu(p,aligner.alignerInfo());
         System.out.println("\nAlignment-Pfad:");
         System.out.println(p.toString());
         System.out.println("\nWagner-Format:");
         System.out.println(p.toWagnerString(m.holeQuelle(),m.holeInfo()));
      }

   }

   /** AlignmentObserver */
   public void statusInformation(String s, boolean erfolg){
      status.setText(s);
   }

}
```

M.5.29 de.denkselbst.mavis.StutzenDialog.java

```
package de.denkselbst.mavis;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import javax.swing.JFrame;
```

```
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JSlider;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.GridLayout;

/**
  * Definiert einen Dialog, mit dem eine Matrix verkleinert werden kann.
  * Wird von Mavis.java aus ins Leben gerufen.
  */
public class StutzenDialog extends JDialog implements ActionListener, ChangeListener{

   private JSlider x;
   private JLabel xl;
   private JSlider y;
   private JLabel yl;
   private JButton ok;
   private JButton cancel;
   private boolean cancelled = false;

   public StutzenDialog(JFrame parent, int maxx, int maxy){
      super(parent, "Matrix verkleinern", true);

      JPanel panel = new JPanel(new GridLayout(2,2));

      x = new JSlider(0,maxx,maxx); x.setSnapToTicks(true);
      y = new JSlider(0,maxy,maxy); y.setSnapToTicks(true);
      x.addChangeListener(this);
      y.addChangeListener(this);
      xl = new JLabel("Breite: "+maxx);
      panel.add(xl);
      panel.add(x);

      yl = new JLabel("Höhe: "+maxy);
      panel.add(yl);
      panel.add(y);

      getContentPane().add(panel,BorderLayout.CENTER);

      ok = new JButton("ok");
      ok.addActionListener(this);
      cancel = new JButton("abbrechen");
      cancel.addActionListener(this);
      JPanel btnPanel = new JPanel(new GridLayout(2,2));
      btnPanel.add(ok);
      btnPanel.add(cancel);

      getContentPane().add(btnPanel,BorderLayout.SOUTH);

      pack();
      show();
   }


   // ActionListener
```

```
   public void actionPerformed(ActionEvent e){
      if(e.getSource() == ok){
         setVisible(false);
      }
      if(e.getSource() == cancel){
         setVisible(false);
         cancelled = true;
      }
   }

   // ChangeListener
   public void stateChanged(ChangeEvent e){
      xl.setText("Breite: "+getXVal());
      yl.setText("Höhe: "+getYVal());
   }

   // x-Wert des Sliders liefern
   public int getXVal(){
      return x.getValue();
   }

   // y-Wert des Sliders liefern
   public int getYVal(){
      return y.getValue();
   }

   public boolean cancelled(){
      return cancelled;
   }

   /** Kleiner Test / Demo */
   public static void main (String[] args){
      StutzenDialog s = new StutzenDialog(null, 20,7);

      if(!s.cancelled()){
         System.out.println("x: "+s.getXVal()+"   y: "+s.getYVal());
      }
   }

}
```

M.5.30 de.denkselbst.mavis.ShowHTMLFrame.java

```
package de.denkselbst.mavis;


import javax.swing.JFrame;
import javax.swing.JEditorPane;
import javax.swing.JScrollPane;
import java.awt.event.WindowListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.Dimension;

/**
  * Diese Klasse öffnet ein HTML-Fenster auf der Oberfläche, das per Konstruktor mit HTML-Code gefüllt wird.
  * Zweck ist das Anzeigen eine Alignments in Textform.
  * Das Fenster ist nach seiner Erzeugung unabhängig von anderen Programmteilen und wird von keinem
  * Objekt kontrolliert. Es reagiert lediglich auf das Schließen-Event, das vom Schließen-Knopf im
  * Fensterrahmen gesendet wird.
```

```
  */

public class ShowHTMLFrame extends JFrame{
   /** Konstruktor */
   public ShowHTMLFrame(String title, String text){
      super(title);

      // JEditorPane zum Anzeigen von HTML benutzen
      JEditorPane htmlpane = new JEditorPane(); // Editor anlegen
      htmlpane.setEditable(false);              // nicht ditierbar
      htmlpane.setContentType("text/html");     // Inhalt = html
      htmlpane.setText(text);                   // Text einsetzen
      htmlpane.setPreferredSize(new Dimension(600,400));

      // Scrollpane um htmlpane wickeln und ins Frame packen
      JScrollPane sp = new JScrollPane(htmlpane);
      sp.setPreferredSize(new Dimension(600,400));
      getContentPane().add(sp);

      // Listener zum Schließen des Fensters
      addWindowListener( new WindowAdapter(){
            public void windowClosing(WindowEvent we){
               setVisible(false);
               dispose();
            }
      });

      pack();
      show();
   }

   /**
     * Kleiner Test
     */
   public static void main(String[] args){
      ShowHTMLFrame sf = new ShowHTMLFrame("TestHTML","<h1>foobar</h1>");
   }
}
```

M.5.31 de/denkselbst/mavis/manifest

```
Main-Class: de.denkselbst.mavis.Mavis
```

Anhang N

Phrasenalignment – Code

N.1 Python-Paket Phrase

N.1.1 __init__.py

```
#!/usr/bin/python

# Initialisierung des Pakets "Region"

__all__ = [
            "zuordnung.py",
]
```

N.1.2 getsegs.py

Sucht Segmente einer bestimmten Länge in der Datenbank.

```
#!/usr/bin/python

import sys, string
from DatabaseAPI import db, config, segmentdb

def get_segs(dbconn, limit, korpora):
    list = dbconn.executelist("select m.segmentnr from map_s_w m, saetze s where m.satzid = s.satzid and s.he
    if list:
        list = dbconn.executelist("select segmentnr, count(tokenid) from map_s_w where segmentnr in (%s) grou
        return list
    return None

def printify(list):
    trans = string.maketrans(")", ",")
    return string.translate(‘list‘, trans, "([]L,")[:-2]


def print_segs(list, dbconn, file):
    textfile = open(file + ".txt", "w")
    segfile = open(file + ".segs", "w")
#    print list
    for id in list:
        segid = db.unknown2string(id[2])
        segfile.write("%s\n" %segid)
        textfile.write("%s:\n" %segid)
        for lang in ("1", "2"):
            segment = segmentdb.SegmentVonDb(segid, dbconn, lang)
            tags = segment.getTags()
            words = segment.getWoerter()
            for(wort, tag) in map(None, words, tags):
```

```
                textfile.write("%s\t%s\n" %(wort, tag))
            if lang == "1":
                textfile.write(10 * "*" + "\n")
        textfile.write(30 * "-" + "\n")

def order_segs(list, count):
    idlist = []
    sum = 0
    segid, swordcount = list[0]
    for row in list:
        (nsegid, wordcount) = row
        if nsegid == segid:
            sum = sum + wordcount
            if swordcount > wordcount:
                swordcount = wordcount
        else:
            idlist.append((swordcount, sum, segid))
            segid, sum = nsegid, wordcount
            swordcount = sum
    idlist.append((swordcount, sum, segid))
    idlist.sort()
    return idlist[0:count]


def main():
    usage = sys.argv[0] + " <segmentgroesse> <segmentanzahl> wb|ko <filename>"
    if len(sys.argv) < 5 or not sys.argv[3] in ("wb", "ko"):
        print usage
        sys.exit(1)
    korpora = " in %s " %config.wbids
    if sys.argv[3] == "ko":
        korpora = " in %s " %config.goodcorpora
#        korpora = " not " + korpora
    file = sys.argv[4]
    dbconn = db.Db()
    list = get_segs(dbconn, sys.argv[1], korpora)
    #    print list
    if list:
        print_segs(order_segs(list, int(sys.argv[2])), dbconn, file)

if __name__ == '__main__':
    main()

```

N.1.3 zuordnung2.py

Versucht, korrespondierende Phrasen zu finden.

```
#!/usr/bin/python

import string
import StringIO
import zuordnung, wb, filterwb
from DatabaseAPI import config
from Region import atom, satz

class OrdneZu:

    def __init__(self, lang1, lang2):
        self.lang1 = lang1
        self.lang2 = lang2
```

```
        self.wb = {}
        self.wb[self.lang1] = wb.LemmatabuchVonDB(self.lang1, self.lang2)
        self.wb[self.lang2] = wb.LemmatabuchVonDB(self.lang2, self.lang1)
        self.tags= {}
        self.tags[self.lang1] = filterwb.tags[config.lang2id[self.lang1]]
        self.tags[self.lang2] = filterwb.tags[config.lang2id[self.lang2]]
        self.tagfolgen = {}
        self.tagfolgen[self.lang1] = self.loadTags(self.lang1, config.tags)
        self.tagfolgen[self.lang2] = self.loadTags(self.lang2, config.tags)
        self.info = StringIO.StringIO()

    def loadTags(self, lang, dir):
        file = open("%s/tags_%s.txt" %(dir, lang), "r")
        tagfolgen = {}
        cat = ""
        hash = {}
        for line in file.readlines():
            line = string.strip(line)
            pos = string.find(line, "###***")
            if pos > -1:
                if cat:
                    tagfolgen[cat] = hash
                cat = line[pos + 6:-1]
                hash = {}
                continue
            if not cat:
                continue
            tags = string.split(line, " ")
            for tag in tags:
                arr = []
                if hash.has_key(tag):
                    arr = hash[tag]
                arr.append(line)
                hash[tag] = arr
        tagfolgen[cat] = hash
        file.close()
        return tagfolgen

    def getInfo(self):
        self.info.seek(0)
        return self.info.read()

    def ordne_zu(self, phrase1, phrase2):
        self.info = StringIO.StringIO()
        ph1 = phrase1.getRegionenMit(atom.Atom)
        ph2 = phrase2.getRegionenMit(atom.Atom)
        kristalle1, kristalle2 = self.dropKnown(ph1, self.lang1, ph2, self.lang2)
        tagfolgen1, rest1 = self.findTagfolgen(kristalle1, self.lang1, ph1)
        tagfolgen2, rest2 = self.findTagfolgen(kristalle2, self.lang2, ph2)
        phrasen = self.ordneTagfolgenZu(tagfolgen1, tagfolgen2)
        phrasen.append((rest1, rest2))
        return phrasen

    def dropKnown(self, phrase1, lang1, phrase2, lang2):
        # phrase1 & 2 sind jeweils Regionen mit Atom
        crystals1, crystals2 = [], []   # Liste von (index, atom, crystal) Paaren
        #es ist markiert, ob etwas uebersetzbar war oder nicht - nur unuebersetzbaren wird zugeordnet
        found1, found2 = {}, {}
        for (lang, phrase, crystals, otherphrase, found, ofound) in [
            (lang1, phrase1, crystals1, phrase2, found1, found2),
            (lang2, phrase2, crystals2, phrase1, found2, found1)
```

```
        ]:
            wb = self.wb[lang]
            irrel = self.tags[lang]        # Bewertung der Einzeltags
            for i in range(len(phrase)):
                patom = phrase[i]
                hypo = (i, patom, 0)
                uehypo = (i, patom, 1)
                tag = patom.getTags()[0]
                lemma = patom.getLemmata()
                wort = patom.getWoerter()[0]
                if found.has_key(wort):
                    crystals.append(uehypo)
                if irrel.has_key(tag) and irrel[tag] < 1:
                    continue
                if lemma[0] == "<unknown>":
                    if "<unknown>" in otherphrase.getLemmata() and wort in otherphrase.getWoerter():
                        # warum reicht nicht die zweite Bedingung?
                        # weil das Wort in der andere Phrase womoeglich nicht unknown ist...
                        # lass uns aber nochmal drueber reden!
                        ofound[wort] = "found"
                        continue
                if not wb.has_key(lemma):
                    crystals.append(hypo)
                    continue
                uebers = wb[lemma]
                if len(uebers) == 0:
                    crystals.append(hypo)
                    continue
                else:
                    wfound = 0
                    otherlems = string.join(otherphrase.getLemmata())
                    for (x, eintrag, uebliste) in uebers:
                        if len(eintrag.getWoerter()) > 1:
                            continue
                        for ueb in uebliste:
                            uelems = string.strip(string.join(ueb.getLemmata()))
                            #todo: Wortgrenzen!!!
                            if uelems and string.find(otherlems, uelems) > -1:
                                ofound[ueb.getWoerter()[0]] = "found"
                                wfound = 1
                                self.info.write("%s -> %s\n" %(‘lemma‘, ‘ueb.getLemmata()‘))
                                break
                    if not wfound:
                        crystals.append(hypo)
                    else:
                        crystals.append(uehypo)
        return crystals1, crystals2

    def p_findTagfolge(self, tagfolge, alltags, pos):
        #anzahl der leerzeichen in alltags bis pos = phrasenanfang
        #anzahl der leerzeichen in tagfolge + start = phrasenende
        start = string.count(alltags, " ", 0, pos)
        end = start + string.count(tagfolge, " ") + 1
        return (start,end)

    def findTagfolgen(self, kristall, lang, phrase):
        tagfolgen = {}
        rest = []
        ourtags = self.tagfolgen[lang]
        slices = {}
        for (pos, patom, ueb) in kristall:
```

```
            found = 0
            tag = patom.getTags()[0]
            min, max = pos - 6, pos + 6
            if min < 0:
                min = 0
            if max >= len(phrase):
                max = len(phrase) - 1
            tagrel = " " + string.join(phrase[min:max].getTags()) + " "
            for kat in ourtags.keys():
                if not ourtags[kat].has_key(tag):
                    continue
                poss_tags = ourtags[kat][tag]
                for ptag in poss_tags:
                    tpos = string.find(tagrel, " " + ptag + " ")
                    if tpos > -1:
                        arr = []
                        if tagfolgen.has_key(kat):
                            arr = tagfolgen[kat]
                        start, end = self.p_findTagfolge(ptag, tagrel, tpos)
                        if end < pos or start > pos or slices.has_key((start,end)):
                            continue
                        found = 1
                        start = start + min
                        end = end + min
                        arr.append((phrase[start:end], ueb))
                        slices[(start,end)] = "found" # eine Tagfolge in 2 Kats???
                        tagfolgen[kat] = arr
            if not found:
                rest.append(patom)
        return tagfolgen, satz.Satz(rest)

    def ordneTagfolgenZu(self, folge1, folge2):
        phrasen = []
        for kat in folge1.keys():
            if folge2.has_key(kat):
                phrasen = phrasen + self.multiply(folge1[kat], folge2[kat])
        return phrasen

    def multiply(self, list1, list2):
        nlist = []
        for (i, ueb1) in list1:
            for (j, ueb2) in list2:
                if not (ueb1 and ueb2): #es darf nicht beides bekannt sein
                    nlist.append((i, j))
        return nlist

```

N.1.4 `tags_de.txt`

Deutsche Tagfolgen

```
###***ADVP:
ADV
ADJD
ADV ADV
ADV ADJD
ADJD ADJD
ADJD ADV
ADV ADV ADJD
ADV ADV ADV
PTKA ADJD
```

```
ADV ADJD ADJD
ADJD ADV ADV
ADJD ADV ADJD
ADV ADV ADV ADJD
ADV ADJD ADV
###***AP:
ADJA
PIDAT
ADV ADJA
ADJD ADJA
ADV PIDAT
ADV ADV ADJA
APPR ART NN ADJA
APPRART NN ADJA
APPR NN ADJA
ADV ADJD ADJA
APPR NE ADJA
ADJD ADJD ADJA
PAV ADJA
APPR ADJA NN ADJA
###***NP:
ART NN
ADJA NN
PPER
ART ADJA NN
PRF
PRELS
PIS
PPOSAT NN
ART NN ART NN
PDS
PDAT NN
PIAT NN
ART NE
PIDAT NN
ART NN NE
ART NN ART ADJA NN
PPOSAT ADJA NN
PWS
ART ADJA NN ART NN
ADV ADJA NN
ADJA NE
ART ADJA ADJA NN
ADJA NN CARD
ART CARD NN
ADJA ADJA NN
ADJA NN ART NN
ART ADJA NN NE
ADJD ADJA NN
ADJA NN NE
PIAT ADJA NN
ART PIDAT NN
ART NN NE NE
ART ADV ADJA NN
ART ADJD ADJA NN
NN
ART NN KON ART NN
ART TRUNC NN
ART TRUNC KON NN
ART ADJA NE
ART ADJA NN NE NE
```

```
ADJA NE NE
PRELAT NN
ART ADJA NN ART ADJA NN
ADJA KON ADJA NN
ART NN PPOSAT NN
ART NN ADJA NN
PDAT ADJA NN
ART NN ART NE
ART NN CARD
ART NN ART NN NE
ART NE NE
ART ADJA CARD NN
ADV PIDAT NN
PIDAT ADJA NN
ADJA NN KON ADJA NN
ART ADJA KON ADJA NN
ADV ADV ADJA NN
ADJA NN ART ADJA NN
ADJA NN NE NE
ART NN PDAT NN
ADJA NN ADJA NN
PPOSAT NN NE
PWAT NN
PPOSAT NN NE NE
ART ADV CARD NN
ART NN KON ART ADJA NN
NE
ART NN ART NN ART NN
ART ADJA TRUNC KON NN
ART APPR ART NN ADJA NN
ADJA TRUNC KON NN
ART ADJA NN KON ART ADJA NN
ART NN NN
ART APPR NN ADJA NN
ART APPRART NN ADJA NN
###***PP:
APPR ART NN
APPRART NN
APPR NN
APPR NE
APPR ART ADJA NN
PAV
APPR ADJA NN
APPRART ADJA NN
APPR PPOSAT NN
APPR ART NN ART NN
APPR NN KON NN
APPR PDAT NN
APPRART NN ART NN
PWAV
APPR PRELS
APPR NE NE
APPR PIS
APPR ART NE
APPR CARD
APPR PIDAT NN
APPR NN ART NN
APPR PPER
APPR ART NN ART ADJA NN
APPR ART NN NE
APPR PPOSAT ADJA NN
```

```
APPRART NN NE
APPRART NE
APPR NN NE
APPR PIAT NN
APPR CARD NN
APPR ART ADJA NN ART NN
APPR PRF
APPRART NN ART ADJA NN
APPR NE KON NE
APPRART NN CARD
APPR ART CARD NN
APPR ART ADJA ADJA NN
APPR NN ART ADJA NN
APPR ART NN CARD
APPRART NN ADV
APPR TRUNC KON NN
APPR NN CARD
APPR NN NE NE
APPR ART ADJA NN NE
APPR ART ADV ADJA NN
APPR ADJA ADJA NN
APPR ADJA NN KON NN
APPR ART ADJD ADJA NN
APPRART ADJA NN CARD
APPR ART PIDAT NN
APPRART ADJA NN ART NN
APPR ADV ADJA NN
APPR ART ADJA CARD NN
APPR ART TRUNC NN
APPR ART NN KON NN
APPRART ADJA NE
APPR ADJD ADJA NN
APPR NN KON ADJA NN
APPR ART NN KON ART NN
APPRART NN KON NN
APPR ART ADJA NE
APPR ADJA KON ADJA NN
APPR ART TRUNC KON NN
APPR PDAT ADJA NN
APPR ART NN PPOSAT NN
APPR ART ADJA NN ART ADJA NN
APPR ADJA NN ART NN
APPRART PIS
APPRART ADJA NN NE
APPR ART NN NE NE
APPR PRELAT NN
APPR ADJA NN KON ADJA NN
APPR ART NN ART NE
APPR PDS
APPR ART NN APPRART NN
APPRART TRUNC KON NN
APPR PIDAT ADJA NN
APPRART NN PPOSAT NN
APPRART TRUNC NN
###***VC:
VVFIN
VAFIN
VVPP
VVINF
VMFIN
VVPP VAFIN
```

```
PTKZU VVINF
VVINF VMFIN
VVPP VAINF
VVIZU
VVIMP
VVPP VAPP
VVPP VAINF VMFIN
VAINF
VAFIN VVPP
VAPP
VVINF VMINF
VVINF VAFIN
PTKZU VVINF VAFIN
VVPP VAPP VAFIN
VVINF VVINF
VVINF PTKZU VMINF
VVINF VVFIN
VAINF VMFIN
PTKZU VAINF
VVPP PTKZU VAINF
VMFIN VVINF
VAPP VAFIN
PTKZU VVINF VVFIN
VVINF PTKZU VVINF
VMFIN VVPP VAINF
###***VP:
          VVFIN
          VAFIN
          VVPP
          VVINF
          VMFIN
           ART NN VVFIN
           VVFIN PPER
           VVFIN ART NN
           VVPP VAFIN
           PTKZU VVINF
           ART NN VAFIN
           VVFIN PRF
           VAFIN ART NN
           PPER VVFIN
           VAFIN PPER
           ART NN VVPP
           VVINF VMFIN
           VVPP VAINF
           PPER VAFIN
           APPR ART NN VVFIN
           ART ADJA NN VVFIN
           ADJA NN VVFIN
           VVIZU
           ART NN VVINF
           APPRART NN VVFIN
           VMFIN ART NN
           ART ADJA NN VAFIN
           VVFIN ART ADJA NN
           ART NN VMFIN
           APPR ART NN VVPP
           VMFIN PPER
           APPR NN VVFIN
           ADJA NN VAFIN
           VAFIN PRF
           PAV VVFIN
```

```
VVIMP
APPR ART NN VAFIN
PDS VVFIN
VVFIN APPR ART NN
PDS VAFIN
VAFIN ART ADJA NN
APPR NE VVFIN
VVPP VAPP
ART NN PTKZU VVINF
VVFIN APPRART NN
ART NN VVPP VAFIN
PPER VMFIN
VVPP VAINF VMFIN
VAINF
ART ADJA NN VVPP
ADJA NN VVPP
VAFIN APPR ART NN
APPR ART NN VVINF
VAFIN VVPP
VAFIN APPRART NN
 APPR ART ADJA NN VVFIN
 APPR NN VVPP
 VMFIN PRF
 PAV VAFIN
 VVFIN APPR NN
 APPRART NN VAFIN
 ART NN ART NN VVFIN
 APPRART NN VVPP
 VVFIN PAV
 VVFIN ADJA NN
 VVFIN PIS
 PPOSAT NN VVFIN
 PIS VVFIN
 APPR NE VAFIN
 APPR NN VAFIN
 APPR ADJA NN VVFIN
 ART ADJA NN VVINF
 VAPP
 APPR NN VVINF
 PDAT NN VVFIN
 VAFIN APPR NN
 VVINF VMINF
 VVINF VAFIN
 ADJA NN VVINF
 VVFIN ART NN ART NN
 VMFIN PIS
 APPR ART ADJA NN VAFIN
 PDAT NN VAFIN
 APPR ART NN VVPP VAFIN
 APPRART ADJA NN VVFIN
 APPR NE VVPP
 APPR ART NN PTKZU VVINF
 VAFIN PAV
 ART NN ART NN VAFIN
 VVFIN PPOSAT NN
 APPR ART ADJA NN VVPP
 APPRART NN VVINF
 ART ADJA NN VMFIN
 VAFIN PIS
 ADJA NN VMFIN
 ART NN VVINF VMFIN
```

```
316             VVFIN APPR ART ADJA NN
317             VAFIN ADJA NN
318             ART NN VVPP VAINF
319             ART NN VVIZU
320             PTKZU VVINF VAFIN
321             VAFIN ART NN ART NN
322             VAFIN PDS
323             ART NN APPR ART NN VVFIN
324             VVPP VAPP VAFIN
325             VAFIN APPR ART ADJA NN
326             PAV VMFIN
327             APPR ADJA NN VVPP
328             PPOSAT NN VAFIN
329             VVFIN PPER APPR ART NN
330             VVFIN ART NN APPR ART NN
331             VVFIN APPR NE
332             ART NE VVFIN
333             ADJA NN PTKZU VVINF
334             VMFIN ART ADJA NN
335             ADJA NN VVPP VAFIN
336             PIS VAFIN
337             APPR ART NN VMFIN
338             PPOSAT NN VVINF
339             VVINF VVINF
340             APPR ART NN ART NN VVFIN
341             VVFIN APPR ADJA NN
342             APPR NN VVPP VAFIN
343             PPOSAT NN VVPP
344             ART ADJA NN PTKZU VVINF
345             VAFIN PPOSAT NN
346             ART NN ART NN VVPP
347             PIDAT NN VVFIN
348             APPR PDAT NN VVFIN
349             PAV VVPP
350             APPR ADJA NN VAFIN
351             PIAT NN VVFIN
352             ART NN APPRART NN VVFIN
353             VVINF PTKZU VMINF
354             ART ADJA NN VVPP VAFIN
355             APPRART ADJA NN VAFIN
356             PIS VMFIN
357             APPR NN PTKZU VVINF
358             APPRART NN VVPP VAFIN
359             APPR ART ADJA NN VVINF
360             VVFIN APPRART ADJA NN
361             VAFIN ART NN APPR ART NN
362             VVFIN PDS
363             VVFIN ART NN APPRART NN
364             VAFIN APPR ADJA NN
365             VVINF VVFIN
366             PDS VMFIN
367             APPR PPOSAT NN VVFIN
368             VAINF VMFIN
369             PTKZU VAINF
370             VVFIN ART NE
371             PAV VVINF
372             PPER VVINF
373             APPRART NN PTKZU VVINF
374             APPR NE VVINF
375             VMFIN APPR ART NN
376             VVPP PTKZU VAINF
```

```
VAFIN APPR NE
VAFIN PIAT NN
APPR PDAT NN VAFIN
PIS VVINF
PIDAT NN VAFIN
VVFIN PIAT NN
ART NN NE VVFIN
ART NN APPR ART NN VVPP
VVFIN ART NN APPR NN
ART NN APPR NE VVFIN
APPR ART NN VVPP VAINF
VVFIN PRF APPR ART NN
VVFIN ART NN NE
PPER VVPP
ART NE VAFIN
APPRART NN VMFIN
ART NN ART ADJA NN VVFIN
APPR ART NN VVINF VMFIN
APPRART NN ART NN VVFIN
VVFIN PPER APPRART NN
APPR ADJA NN VVINF
VAFIN APPRART ADJA NN
APPR ART NN ART NN VVPP
APPR NN VMFIN
VVFIN APPR PPOSAT NN
APPR NN KON NN VVFIN
ART NN APPR NN VVFIN
VVFIN PIDAT NN
VMFIN PAV
ART NN VAFIN VVPP
APPR NN VVPP VAINF
VMFIN VVINF
VMFIN APPR NN
PIAT NN VVINF
VAPP VAFIN
VAFIN ART NN APPR NN
ART NN ART NN VVINF
PPOSAT ADJA NN VVFIN
APPR ART NN VVIZU
ADJA NN VVPP VAINF
PRF VVINF
PDAT NN VMFIN
PIAT NN VAFIN
VMFIN APPRART NN
ART NN VVPP VAPP
APPR NE VVPP VAFIN
APPR ART ADJA NN VMFIN
APPR ART ADJA NN VVPP VAFIN
PIS VVPP
ART NN APPR ART NN VAFIN
ART ADJA NN VVINF VMFIN
ART ADJA NN ART NN VVFIN
ART ADJA NN VVIZU
VVFIN ART NN ART ADJA NN
PTKZU VVINF VVFIN
PWS VVFIN
APPR PPER VVFIN
APPRART ADJA NN VVPP
VVINF PTKZU VVINF
ART NN VVPP VAINF VMFIN
VAFIN PDAT NN
```

```
        ART NN APPR ART NN VVINF
        ART NN APPR NE VAFIN
        ART ADJA NN VVPP VAINF
        PRF VVFIN
        APPR ADJA NN VVPP VAFIN
        VVFIN PRF APPRART NN
        PRELS VVFIN
        PIAT NN VVPP
        APPR PPOSAT NN VVPP
        VAFIN ART NN APPRART NN
        APPR NN VVINF VMFIN
        VMFIN VVPP VAINF
        VAFIN PPER APPR ART NN
        VVFIN PDAT NN
        ADJA NN VVIZU
        PPOSAT NN PTKZU VVINF
        ADJA NN VVINF VMFIN
        VVFIN APPR ART NN ART NN
        APPR ART NN ART NN VAFIN
        VAFIN ART NE
        APPR NN ART NN VVFIN
        ART NN APPRART NN VAFIN
        ART NN APPR ART ADJA NN VVFIN
        APPR NE VMFIN
        VAFIN APPR PDAT NN
        ART NN NE VAFIN
        ART NN APPR NN VVPP
        APPR NE NE VVFIN
        PPOSAT NN VVPP VAFIN
        ADV ADJA NN VVFIN
        VVFIN ART NN APPR NE
        VVFIN APPR PDAT NN
        APPRART NN VVPP VAINF
        APPR ART NE VVFIN
        APPR ART ADJA NN PTKZU VVINF
        VVFIN PPER APPR NN
        VVFIN ART NN NE NE
        VMFIN PPOSAT NN
        PWS VAFIN
        APPR NN KON NN VVPP
        VAFIN ART NN ART ADJA NN
        VMFIN ADJA NN
        APPR PPOSAT NN VVINF
        VVFIN ART NN APPR ART ADJA NN
        VMFIN APPR ART ADJA NN
        VAFIN PIDAT NN
        VAFIN APPR PPOSAT NN
        ART NN ART ADJA NN VAFIN
        VVFIN PRF APPR NN
        VVFIN PRF PAV
        ART NN APPRART NN VVPP
        APPR ADJA NN PTKZU VVINF
        VAFIN APPR ART NN ART NN
        VAFIN ADV ADJA NN
        PWAV VVFIN
        VVFIN PPER PAV
        APPR ART NN ART NN VVINF
        VAFIN ART NN APPR NE
        PIS PTKZU VVINF
        ART PIDAT NN VVFIN
        PRF VVPP
```

```
499             ART NN ART NN VMFIN
500             ART NN ART NN PTKZU VVINF
501             APPR NN VVIZU
502             APPR NN KON NN VAFIN
503             APPR PIDAT NN VVFIN
504             PIDAT NN VVPP
505             ART NN APPR NN VAFIN
506             APPR ADJA NN VMFIN
507             ART ADJA ADJA NN VVFIN
508             APPR PPOSAT NN VAFIN
509             VVFIN ADV ADJA NN
510             APPRART NN ART NN VVPP
511             VAFIN APPR PIS
512             APPR PPER VAFIN
513             VVFIN PPER APPR ART ADJA NN
514             VMFIN ART NN ART NN
515             VVFIN ART ADJA NN APPR ART NN
516             ART ADJA NN APPR ART NN VVFIN
517             APPR NE PTKZU VVINF
518             APPR ART NN VVPP VAPP
519             PDAT NN VVPP
520             APPR NN ART NN VAFIN
521             APPRART NN VVINF VMFIN
522             VVFIN APPR PIS
523             VAFIN ART NN APPR ART ADJA NN
524             VAFIN APPR NN ART NN
525             ART NN ART NN VVPP VAFIN
526             VMFIN ART NN APPR ART NN
527             PPER VAFIN VVPP
528             APPR ART ADJA NN VVPP VAINF
529             VAFIN ART ADJA NN ART NN
530             APPR ART NN ART ADJA NN VVFIN
531               # Corpus:  tut-fr (Auszug aus: Frankfurter Rundschau (Juli  - März ))
532               # Size:     intervals/matches
533               # Size:     intervals/matches
534               # Size:     intervals/matches
535               # Size:     intervals/matches
536               # Size:     intervals/matches
537               # Size:     intervals/matches
538               # Size:     intervals/matches
539               # Size:     intervals/matches
540               # Date:    Wed Jul  ::
541               # Date:    Wed Jul  ::
542               # Date:    Wed Jul  ::
543               # Date:    Wed Jul  ::
544               # Date:    Wed Jul  ::
545               # Date:    Wed Jul  ::
546               # Date:    Wed Jul  ::
547               # Date:    Wed Jul  ::
```

N.1.5 `tags_en.txt`

Englische Tagfolgen

```
1   ###***NP:
2             NN
3             DT NN
4             JJ NN
5             DT JJ NN
6              NNS
7              DT NNS
```

```
        NP
        DT NP
        NN NN
        DT NN NN
        NP NP
        DT NP NP
        JJ NNS
        DT JJ NNS
        JJ NN NN
        DT JJ NN NN
        NP NN
        DT NP NN
        NN NNS
        DT NN NNS
        NP NP NP
        DT NP NP NP
        JJ JJ NN
        DT JJ JJ NN
         DT CD NNS
         CD NNS
         NN NN NN
         DT NN NN NN
         JJ NP
         DT JJ NP
         JJ NP NP
         DT JJ NP NP
         JJ JJ NNS
         DT JJ JJ NNS
         JJ NN NNS
         DT JJ NN NNS
         NP NN NN
         DT NP NN NN
         NP NNS
         DT NP NNS
         NP NPS
         DT NP NPS
         NPS
         DT NPS
         JJ CD NNS
         DT JJ CD NNS
         DT CD NN
         CD NN
         JJ JJ NN NN
         DT JJ JJ NN NN
         JJR NN
         DT JJR NN
         NP NP NN
         DT NP NP NN
         NNS NN
         DT NNS NN
         JJ NP NN
         DT JJ NP NN
         JJ NN NN NN
         DT JJ NN NN NN
         JJS NN
         DT JJS NN
         DT CD NN NN
         CD NN NN
         NN NP
         DT NN NP
         NN NP NP
```

```
            DT NN NP NP
            JJ NN NP
            DT JJ NN NP
            RB JJ NN
            NP NP NP NP
            NN NN NNS
            DT RB JJ NN
            DT NP NP NP NP
            DT NN NN NNS
            NN NN NP
            DT NN NN NP
            DT CD JJ NNS
            CD JJ NNS
            JJ NP NP NP
            DT JJ NP NP NP
            JJS NNS
            DT JJS NNS
###***PP:
        IN DT NN
        IN NP
        IN NN
        IN DT JJ NN
        IN DT NP
        IN JJ NN
        IN NNS
        IN DT NN NN
        IN JJ NNS
        IN DT NNS
        IN NP NP
        IN NN NN
        TO DT NN
        IN DT NP NP
        IN CD NN
        IN NN NNS
        IN DT NP NN
        TO NP
        TO NN
        IN CD NNS
        IN DT JJ NN NN
        IN DT JJ NNS
        IN NP NN
        IN DT JJ JJ NN
        IN JJ NN NN
        TO DT NP
        TO DT JJ NN
        IN DT JJ NP
        IN NP NP NP
        IN DT NP NP NP
        TO JJ NN
        TO CD NN
        IN JJ JJ NN
        IN DT CD NN
        IN JJ NN NNS
        IN DT NN NNS
        TO NNS
        IN JJ NP
        TO NP NP
        IN DT NN NN NN
        IN CD CD NN
        TO DT NN NN
        TO DT NNS
```

```
IN NP NNS
TO JJ NNS
IN JJ JJ NNS
IN JJR NN
IN DT NP NN NN
TO CD NNS
IN DT CD NNS
IN NN NN NN
TO DT NP NP
TO NN NN
IN DT JJ NP NP
IN DT JJ JJ NNS
IN DT JJ CD NN
IN CD CD NNS
IN DT NP NNS
IN DT JJ CD NNS
IN DT NPS
IN DT NP NPS
IN CD JJ NN
IN NPS
IN DT JJ NN NNS
IN NP NP NP NP
IN NN NP
IN DT JJ NP NN
IN RB CD NN
IN CD NP
IN DT NN NP
IN CD JJ NNS
IN NN NN NNS
TO DT NP NN
TO NN NNS
IN DT JJ JJ NN NN
IN JJR NNS
IN NP NPS
IN DT JJ NN NN NN
IN DT NP NP NN
IN DT CD NN NN
IN NP NN NN
TO DT JJ NN NN
IN DT NNS NN
IN DT JJR NN
IN JJ NP NP
IN DT JJS NN
IN NNS NN
IN NP NP NN
IN JJ CC JJ NN
IN DT JJ NN NP
IN CD NN NN
IN JJ NP NN
TO NP NP NP
TO JJ NN NN
TO DT JJ NNS
IN RB CD NNS
IN RB JJ NN
TO NP NN
IN JJ JJ NN NN
IN DT CD JJ NN
IN DT NP NP NP NP
IN DT NN NN NP
TO CD CD NN
IN CD CC CD NN
```

```
###***VC:
            VBD VBN
            VB VBN
            VBZ VBN
            VBP VBN
             VBZ TO VB
             VBD TO VB
             VBD RB VBN
             VBZ VBG
             VBP TO VB
             VB TO VB
             VBD VBN VBN
             VBP VBG
             VBZ RB VBN
             VBZ VBN VBN
             VBD VBG
             VBD RB VB
             VBG VBN
             VB VBG
             VBP VBN VBN
             VBP RB VBN
             VBZ RB VB
             VB JJ TO VB
             VBN VBN
             VBG TO VB
             VBZ TO VB VBN
             VBD TO VB VBN
             VBN TO VB
             VBP RB VB
             VBP TO VB VBN
             VBD JJ TO VB
             VBZ RB VBG
             VBP RB VBG
###***VP:
          VBD VBN
          VB VBN
          VBZ VBN
          VBP VBN
          VBN VBN
          VBZ TO VB
          VBD RB VB
          VBN TO VB
          VBD TO VB
          VBZ RB VB
          VBD RB VBN
          VBP TO VB
          VB TO VB
          VBZ VBG
          VBP RB VB
          VBZ RB VBN
          VBP VBG
          VBG VBN
          VBD VBN VBN
          VBD VBG
          VBZ VBN VBN
          VBG TO VB
          VBZ VBN DT NN
          VBP RB VBN
          VBD VBN DT NN
          VB VBG
          VBP VBN VBN
```

```
        VBD VBN NN
        VB JJ TO VB
        VBZ TO VB DT NN
        VBZ TO VB VBN
        VBN TO VB DT NN
        VBD TO VB VBN
        VBZ TO VB NN
        VBZ VBN DT JJ NN
        VBD TO VB DT NN
        VBZ VBN NN
        VBZ RB VBG
        VBP VBN DT NN
        VBD JJ TO VB
        VBN TO VB NN
        VBP TO VB VBN
        VBP RB VBG
        VBP TO VB DT NN
        VB TO VB DT NN
        VBD RB VB DT NN
        VBD TO VB NN
        VBD VBN DT JJ NN
        VBD VBN JJ NN
        VB VBN DT NN
        VBP VBN NN
        VBZ TO VB DT JJ NN
        VBP TO VB NN
        VB VBN NN
        VBZ VBN JJ NN
        VB TO VB NN
        VBZ VBG NN
        VBZ VBG DT NN
        VB VBG NN
        VBD VBN NNS
        VBZ TO VB JJ NN
        VBZ VBN NNS
```

N.1.6 `phrasealign.py`

Benutzt `zuordnung2.py`, um Kollokationskandidaten zu finden, und trägt diese in die Datenbank ein.

```
#!/usr/bin/python

#TODO:
#uebriggebliebene zuordnen
#Tabellen
#eintragen mit belegen
import sys, string
from DatabaseAPI import db, segmentdb, config, regiondb, qclient
from Region import atom, satz
from Phrase import zuordnung2
import filterwb, abstand

class PhraseAlign:

    def __init__(self, db, file, opt_enterdb = 1):
        self.opt_enterdb = opt_enterdb
        self.db = db
        regiondb.setDefaultDbconn(db)
        self.segments = file.readlines()
        self.satzidx = self.db.getMaxIndex("satzid", "saetze")
        self.segmentidx = self.db.getMaxIndex("segmentnr", "map_s_w")
```

```
        self.mapidx = self.db.getMaxIndex("mapid", "map_s_w")
        self.pe_idx = self.db.getMaxIndex("pe_id", "phrase_examples")
        self.pc_idx = self.db.getMaxIndex("pc_id", "phrase_count")
        self.zuord = zuordnung2.OrdneZu("en", "de")
        self.dbq = qclient.Query()
        if not self.dbq.ok():
            sys.stderr.write("Query-Server nicht gefunden!\n")
        self.dbq.seterkkorpus()
        file.close()

    def align(self):
        for segment in self.segments:
            segment = string.strip(segment)
            list = self.db.executelist("select segmentnr from phrase_examples where segmentnr = %s" %segment)
            if list:
                sys.stderr.write("segment %s already aligned!\n" %segment)
                continue
            phrase_de = segmentdb.SegmentVonDb(segment, self.db,
                                               config.lang2id["de"])
            phrase_en = segmentdb.SegmentVonDb(segment, self.db,
                                               config.lang2id["en"])
            zuords = self.zuord.ordne_zu(phrase_en, phrase_de)
            print phrase_de.getText()
            print phrase_en.getText()
            print 36 * "*-"
            for (p_en, p_de) in zuords:
                self.print_segs(p_en, p_de)
                if self.opt_enterdb:
                    self.enter(p_de, p_en, segment)
            print self.zuord.getInfo()

    def print_segs(self, phrase1, phrase2):
        print phrase1.getWoerter(), phrase1.getTags()
        print phrase2.getWoerter(), phrase2.getTags()
        print 72 * "#"

    def enter(self, phrase_de, phrase_en, belegid):
        phrase_match_de = phrase_de.getWoerter()
        phrase_match_en = phrase_en.getWoerter()
        if len(phrase_match_de) > len(phrase_match_en):
            lang = "de"
            phrase_match = string.join(phrase_match_de)
        else:
            lang="en"
            phrase_match = string.join(phrase_match_en)
        if not phrase_match:
            return
        #print "\tabgefragte Woerter:", phrase_match
        segmente = self.dbq.segment(match_f = phrase_match, strict = "true", lang = lang)
        #print "\tgefundene Segmente:", segmente
        equ = abstand.AbstandEQU()
        gefunden = 0
        for segment in segmente:
            if lang == "de":
                segm_de = segment
                segm_en = segment.alignedRegion()
            else:
                segm_en = segment
                segm_de = segment.alignedRegion()
            #print phrase_de.getTags(), segm_de.getTags()
            #print phrase_en.getTags(), segm_en.getTags()
```

```
83              #print "de_equ", equ(phrase_de, segm_de)
84              #print "en_equ", equ(phrase_en, segm_en)
85              gefunden = (equ(phrase_de, segm_de) < 0.5) and (equ(phrase_en, segm_en) < 0.5)
86              if gefunden:
87                  break
88          if gefunden:
89              #print "\tbenutztes Segment:", segment.getWoerter()
90              segid = db.unknown2string(segment.regionid)
91              list = self.db.executelist(
92                  ("select pc_id, zaehler ") +\
93                  ("from phrase_count ") +\
94                  ("where segmentnr = %s" %(segid))
95              )
96              if list:
97                  pc_idx, zaehler = list[0]
98                  zaehler = self.db.incr(zaehler)
99                  self.db.update(
100                     ("update phrase_count ") +\
101                     ("set zaehler = %s " %zaehler) +\
102                     ("where pc_id = %s" %db.unknown2string(pc_idx))
103                 )
104         else: #nicht gefunden
105             langs = {"de" : phrase_de, "en": phrase_en}
106             for i in langs.keys():
107                 lang_id = config.lang2id[i]
108                 self.db.update(
109                     ("insert into saetze ") +\
110                     ("(satzid, sprachenid, herkunftsid) ") +\
111                     ("values(%s, %s, %s)" %(self.satzidx, lang_id, -1))
112                 )
113                 #print "satzid:", self.satzidx
114                 #print "sprache:", lang_id
115                 #print "segment:", self.segmentidx
116                 self.enterphrase(langs[i], self.satzidx, i)
117                 self.satzidx = self.db.incr(self.satzidx)
118             self.db.update(
119                 ("insert into phrase_count ") +\
120                 ("(pc_id, segmentnr, zaehler) ") +\
121                 ("values(%s, %s, 1)" %(self.pc_idx, self.segmentidx))
122             )
123             self.pc_idx = self.db.incr(self.pc_idx)
124             #todo: das folgende schoener machen!!!
125             self.dbq.query("cache.delete t:segment;s:text;w:strict=true,lang=%s,match=%s" %(lang, phrase_matc
126             self.dbq.query("cache.delete t:segment;s:pyob;w:match_f=%s,strict=%s,lang=%s" %(phrase_match, "tr
127         self.db.update(
128             ("insert into phrase_examples ") +\
129             ("(pe_id, pc_id, segmentnr) ") +\
130             ("values(%s, %s, %s)" %(self.pe_idx, self.pc_idx, belegid))
131         )
132         self.segmentidx = self.db.incr(self.segmentidx)
133         self.pe_idx = self.db.incr(self.pe_idx)
134
135     def enterphrase(self, phrase, satzid, lang):
136         all = map(None, phrase.getWoerter(), phrase.getTags(), phrase.getLemmata())
137         for (word, tag, gf) in all:
138 #            print word, tag, gf
139             tokenid = self.db.executelist(
140                 ("select t.tokenid ") +\
141                 ("from tokens t, grundformen gf, tagset_ims_%s tags " %lang) +\
142                 ("where t.name = '%s' " %word) +\
143                 ("and t.grundformid = gf.grundformid ") +\
```

```
                ("and gf.name = '%s' " %gf) +\
                ("and t.tagid = tags.tagid ") +\
                ("and tags.name = '%s' " %tag)
            )
            print tokenid
            if tokenid:
                tokenid = tokenid[0][0]
                    self.db.update(
                        ("insert into map_s_w ") +\
                        ("(mapid, tokenid, satzid, segmentnr) ") +\
                        ("values(%s, %s, %s, %s)" %(self.mapidx,
                                        db.unknown2string(tokenid),
                                        satzid,
                                        self.segmentidx
                                       ))
                    )
                    self.mapidx = self.db.incr(self.mapidx)

    def close(self):
        self.db.close()
        self.dbq.close()

def usage():
    print "Usage: phrasealign.py [-n] [<file with segment-IDs>]"
    print "-n        do not enter stuff into database"
    sys.exit()

def main():
    import getopt
    # process command line args
    try:
        opts, args = getopt.getopt(sys.argv[1:], "hn", ["help", "nodb"])
    except getopt.GetoptError:
        # print help information and exit:
        usage()
    opt_enterdb = 1
    for o, a in opts:
        if o in ("-n", "--nodb"):
            opt_enterdb = 0
        if o in ("-h", "--help"):
            usage()
    dbconn = db.Db()
    file = sys.stdin
    if len(args) > 0:
        file = open(args[0], "r")
    paligner = PhraseAlign(dbconn, file, opt_enterdb)
    paligner.align()
    paligner.close()

if __name__ == '__main__':
    main()
```

N.1.7 demo-zuord2.py

```
#!/usr/bin/python

from DatabaseAPI import segmentdb, db, regiondb
from Phrase import zuordnung2
import string
import sys
```

```

def main():
    dbconn = db.Db()
    regiondb.setDefaultDbconn(dbconn)
    reg1 = segmentdb.SegmentVonDb(111219, dbconn, "1")
    reg2 = segmentdb.SegmentVonDb(111219, dbconn, "2")
    print reg1.getText()
    print reg2.getText()
    ord = zuordnung2.OrdneZu("de", "en")
    phrases = ord.ordne_zu(reg1, reg2)
    for (i, j) in phrases:
        print i.getText()
        print j.getText()
        print 72 * "-"
    dbconn.close()

if __name__ == '__main__':
    main()


```

N.1.8 `mogel.py`

Dieses Skript hilft beim manuelle Einfügen von erkannten Phrasen mit Beispielsätzen in die Datenbank.

```
#!/usr/bin/python

import korpus2db

# Eingabedatei: nacheinander immer 3 Segmente: de, en, ID-Liste der Beispiele

import sys
import string

k = korpus2db.Korpus()
f = open(sys.argv[1], "rt")
segliste = string.split(f.read(), "<segmentgrenze>")
tagset_de = k.dbconn.executelist("select name, tagsetid from tagsets where sprachenid = %s and name = '%s'" %
tagset_en = k.dbconn.executelist("select name, tagsetid from tagsets where sprachenid = %s and name = '%s'" %
tagsets=[tagset_de[0], tagset_en[0]]
sprachen = ["de", "en"]
pc_id = k.dbconn.getMaxIndex("pc_id", "phrase_count")
pe_id = k.dbconn.getMaxIndex("pe_id", "phrase_examples")
for i in range(len(segliste)):
    segment = [segliste[i]]
    #sys.stderr.write("segment = %s\n" %`segment`)
    sprache = 1 + (i % 3)
    if sprache == 1:
        segstart = k.segment_idx
        sys.stderr.write("Processing %s -> %s\n" %(i,segstart))
    elif sprache == 2:
        k.segment_idx = segstart
    elif sprache == 3:
        idlistestr = string.split(segment[0])
        idliste = []
        for idstr in idlistestr:
            if idstr:
                idliste.append(int(idstr))
        k.dbconn.update("insert into phrase_count(pc_id, segmentnr, zaehler) values (%s, %s, %s)" %(pc_id, se
        for seg_id in idliste:
            k.dbconn.update("insert into phrase_examples(pe_id, pc_id, segmentnr) values(%s, %s, %s)" %(pe_id
```

```
            pe_id = k.dbconn.incr(pe_id)
        sys.stderr.write("%s Beispiele fuer Phrase %s\n" %(len(idliste),pc_id))
        pc_id = k.dbconn.incr(pc_id)
        continue
    else:
       raise IndexError
    k.tag_idx = k.dbconn.getMaxIndex("tagid", "tagset_ims_%s" %sprachen[sprache-1])
    k.entersegmente(segment, k.sent_idx, -1, sprache, tagsets[sprache-1])
f.close()
k.close()
```

N.1.9 web_phrases.py

Die erkannten Phrasen und ihre Beispiele (Belege) können mit diesem CGI-Skript dargestellt werden. Bei der Installation im cgi-bin Verzeichnis ist zu beachten, dass die Pakete Region und DatabaseAPI verfügbar sein müssen.

```
#!/usr/bin/python

from DatabaseAPI import segmentdb, config, regiondb, db
from Region import atom
import base64
import cgi
import pickle
import socket
import string
import sys
import urllib

# web_phrases.py ist ein CGI-Skript, das die von KoKS slebst erkannten
# Phrasen und die zugehoerigen Beispiele tabellarisch darstellt.
# Joachim Wagner, 24. Nov. 2001

scriptname = urllib.quote('web_phrases.py')

print "Content-type: %s\n" %"text/html"  # Extra-Zeilentrenner wichtig!

def kopf(titel):
    print """<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
    <html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
      <title>%s</title>
    </head>
    <body>
    <h1>%s</h1>
""" %(titel, titel)

def ende():
    print "</body></html>"

def printseg(segdb):
    segdb_str=pickle.dumps(segdb)
    segdb_64=base64.encodestring(segdb_str)
    print '<a href="%s?segment=%s">' %(scriptname, urllib.quote(segdb_64))
    print cgi.escape(segdb.getText())
    print '</a>'

def seg_sort_fun(a,b):
    token_a = dbconn.executelist(
        "select tokenid " +
```

```
        "from map_s_w " +
        "where segmentnr = %s" %a
    )
    token_b = dbconn.executelist(
        "select tokenid " +
        "from map_s_w " +
        "where segmentnr = %s" %b
    )
    return len(token_a) - len(token_b)

dbconn = db.Db()
regiondb.setDefaultDbconn(dbconn)
try:
    form = cgi.FieldStorage()
    form_ok = 0
    if form.has_key("phrsid"):
        phrsid = form["phrsid"].value
        kopf("Beispiele zur Phrase %s" %phrsid)
        orderby = ""
        sortl = 1   # Beispiele nach Laenge sortieren
        try:
            orderby = form["sort"].value
            sortl = 0
            if orderby == "length":
                sortl = 1
                orderby = ""
                print '<p>Sortiert nach Länge.</p>'
            if orderby not in ["segmentnr", "pe_id", "pc_id"]:
                print '<p>Fehler: %s kein gültiges Sortierkriterium</p>' %cgi.quote(orderby)
                orderby = ""
            else:
                print '<p>Sortiert nach %s.</p>' %orderby
                orderby = " order by " + orderby
        except:
            pass
        data = dbconn.executelist(
            "select pe_id, segmentnr " +
            "from phrase_examples " +
            ("where pc_id = %s" %phrsid) + orderby
        )
        print '<table border="1">'
        print '<tr>'
        print '<th><a href="%s?phrsid=%s&sort=pe_id">pe_id</a></th>' %(scriptname,urllib.quote(phrsid))
        print '<th><a href="%s?phrsid=%s&sort=segmentnr">seg_id</a></th>' %(scriptname,urllib.quote(phrsid))
        print '<th>Deutsch</th><th>Englisch</th>'
        print '</tr>'
        if sortl:
            data.sort(lambda x,y: seg_sort_fun(int(x[1]),int(y[1])))
        for (pe_id, segmentnr) in data:
            print '<tr><td align="right">'
            print int(pe_id)
            print '</td><td align="right">'
            print int(segmentnr)
            print '</td><td>'
            printseg(segmentdb.SegmentVonDb(int(segmentnr), dbconn, "1"))
            print '</td><td>'
            printseg(segmentdb.SegmentVonDb(int(segmentnr), dbconn, "2"))
            print '</td></tr>'
        print '</table>'
        if not sortl:
            print '<p><a href="%s?phrsid=%s&sort=length">Nach Länge sortieren.</a></p>' %(scriptname,urllib.q
```

```
        print '<p><a href="%s">Zurück zur Phrasenübersicht</a></p>' %scriptname
    elif form.has_key("segment"):
        segdb_64 = form["segment"].value
        segdb_str = base64.decodestring(segdb_64)
        segdb = pickle.loads(segdb_str)
        kopf("Details zu %s" %cgi.escape(segdb.getText()))
        print '<table border="1">'
        print '<tr><th>Wort</th><th>Tag</th><th>Lemma</th></tr>'
        wtl = map(None, segdb.getWoerter(),
                  segdb.getTags(),
                  segdb.getLemmata())
        for line in wtl:
            print '<tr>'
            for item in line:
                print '<td>%s</td>' %cgi.escape(item)
            print '</tr>'
        print '</table>'
        print '<p><a href="%s">Zurück zur Phrasenübersicht</a></p>' %scriptname
    else:
        kopf("Phrasen mit mindestens zwei Beispielen")
        print "<p>Einzelworteinträge ausgelassen.</p>"
        data = dbconn.executelist(
            "select pc_id, segmentnr, zaehler "
            "from phrase_count "
            "where zaehler > 1 "
            "order by zaehler desc"
        )
        print '<table border="1">'
        print '<tr><th>pc_id</th><th>Beispiele</th><th>Deutsch</th><th>Englisch</th></tr>'
        for (pc_id, segmentnr, zaehler) in data:
            segde = segmentdb.SegmentVonDb(int(segmentnr), dbconn, "1")
            segen = segmentdb.SegmentVonDb(int(segmentnr), dbconn, "2")
            atmde = segde.getRegionenMit(atom.Atom)
            atmen = segen.getRegionenMit(atom.Atom)
            if len(atmde) == 1 and len(atmen) == 1:
                continue
            print '<tr><td>'
            print int(pc_id)
            print '</td><td>'
            print '<a href="web_phrases.py?phrsid=%s">%s</a>' %(int(pc_id),int(zaehler))
            print '</td><td>'
            printseg(segde)
            print '</td><td>'
            printseg(segen)
            print '</td></tr>'
        print '</table>'
    ende()
    dbconn.close()
except socket.error, data:
    kopf("Fehler")
    print socket.error, data
    ende()
    dbconn.close()
```

Anhang O

Datenbank – Code

O.1 SQL-Skripte zum Erstellen der Datenbank

O.1.1 `db.sql`

Diese Skript erzeugt die Datenbank.

```
CREATE TABLE autoren (
        autorid int8 primary key,
        name varchar(100) BINARY);
CREATE TABLE herkunft (
        herkunftsid int8 primary key,
        name varchar(100) BINARY,
        autorid int8 references autoren,
        jahr varchar(30) BINARY,
        quellenid int8 references quellen);
CREATE TABLE quellen (
        quellenid int8 primary key,
        name varchar(100) BINARY,
        verlag varchar(100) BINARY,
        rechte varchar(200) BINARY,
        bemerkungen varchar(200) BINARY,
        statusid int4);
CREATE TABLE tokens (
        tokenid int8 primary key,
        name varchar(50) BINARY not null,
        reverse varchar(50) BINARY not null,
        grundformid int8 not null references grundformen on delete cascade,
        tagid int8 not null,
        tagqualitaet int4,
        tagsetid int8);
CREATE TABLE saetze (
        satzid int8 primary key,
        herkunftsid int8 not null references herkunft on delete cascade,
        parsequalitaet int4,
        segmentqual int4,
        prev int8,
        succ int8,
        sprachenid int8 not null);
CREATE TABLE map_s_w (
        mapid int8 primary key,
        satzid int8 not null references saetze on delete cascade,
        tokenid int8 not null references tokens,
        segmentnr int8 not null);
CREATE TABLE grundformen (
        grundformid int8 primary key,
        name varchar(100) BINARY not null,
```

```
        reverse varchar(100) BINARY not null,
        tagid int8 not null,
        tagsetid int8 not null);
CREATE TABLE phrase_count(
        pc_id int8 primary key,
        segmentnr int8 not null references map_s_w on delete cascade,
        zaehler int8);
CREATE TABLE phrase_examples(
        pe_id int8 primary key,
        pc_id int8 references phrase_count on delete cascade,
        segmentnr int8 not null references map_s_w on delete cascade);
CREATE TABLE sprachen (
        sprachenid int8 primary key,
        name varchar(100) BINARY);
CREATE TABLE tagsets (
        tagsetid int8 primary key,
        name varchar(100) BINARY,
        sprachenid int8);
CREATE TABLE tagset_ims_de (
        tagid int8 primary key,
        name varchar(100) BINARY);
CREATE TABLE tagset_ims_en (
        tagid int8 primary key,
        name varchar(100) BINARY);
CREATE TABLE tagset_lql_de (
        tagid int8 primary key,
        name varchar(100) BINARY);
CREATE TABLE tagset_lql_en (
        tagid int8 primary key,
        name varchar(100) BINARY);
CREATE TABLE status (
        statusid int8 primary key,
        name varchar(100) BINARY);
insert into tagsets (tagsetid, name, sprachenid) values (1, 'tagset_ims_de', 1);
insert into tagsets (tagsetid, name, sprachenid) values (2, 'tagset_ims_en', 2);
insert into tagsets (tagsetid, name, sprachenid) values (3, 'tagset_lql_de', 1);
insert into tagsets (tagsetid, name, sprachenid) values (4, 'tagset_lql_en', 2);
insert into herkunft(herkunftsid, name, autorid, jahr, quellenid) values (-1, "erkannt", -1, null, -1);
create index grundformid_idx on tokens (grundformid);
create index tagid_idx on tokens (tagid);
create index segmentnr_index on map_s_w(segmentnr);
create index satzid_index on map_s_w(satzid);
create index tokenid_index on map_s_w(tokenid);
create index sprachenid_index on saetze(sprachenid);
create index herkunftsid_index on saetze(herkunftsid);
create index tokenname_index on tokens (name);
create index reverse_index on tokens (reverse);
create index gf_reverse_index on grundformen (reverse);
create index grundformname_index on grundformen (name);
create index pe_id_index on phrase_examples(pe_id);
create index pc_segment_index on phrase_count(segmentnr);
create index pc_id_index on phrase_count(pc_id);
create index pe_segment_index on phrase_examples(segmentnr);
```

O.1.2 `index.sql`

Dieses Skript fügt die Indizes hinzu.

```
create unique index tokenid_idx on tokens (tokenid);
create unique index tag_de_idx on tagset_ims_de (tagid);
create unique index tag_en_idx on tagset_ims_en (tagid);
```

```
create unique index autorid_idx on autoren (autorid);
create unique index herkunftsid_idx on herkunft (herkunftsid);
create unique index quellenid_idx on quellen (quellenid);
create unique index mapid_idx on map_s_w (mapid);
create unique index sprachenid_idx on sprachen (sprachenid);
create unique index statusid_idx on status (statusid);
create unique index tagsetid_idx on tagsets (tagsetid);
create unique index satzid_idx on saetze (satzid);
create unique index grundformid_index on grundformen (grundformid);
/*create index grundformid_idx on tokens (grundformid);
create index tagid_idx on tokens (tagid);
create index segmentnr_index on map_s_w(segmentnr);
create index satzid_index on map_s_w(satzid);
create index tokenid_index on map_s_w(tokenid);
create index sprachenid_index on saetze(sprachenid);
create index herkunftsid_index on saetze(herkunftsid);
create index grundformname_index on grundformen (name);
create index tokenname_index on tokens (name);*/
/*create unique index gf_name_tag_index on grundformen (name, tagid);
create unique index tk_name_tag_gf_index on tokens(name, tagid, grundformid);*/
```

O.1.3 `clean.sql`

Mit diesem Skript löscht man aus allen Tabellen, in die beim Eintragen von Korpora geschrieben wird.

```
DELETE from tokens;
DELETE from saetze;
DELETE from map_s_w;
DELETE from autoren;
DELETE from quellen;
DELETE from herkunft;
DELETE from grundformen;
DELETE from tagset_ims_de;
DELETE from tagset_ims_en;
```

O.1.4 `mrproper.sql`

Dieses Skript löscht alle angelegten Tabellen.

```
drop table autoren;
drop table grundformen;
drop table herkunft;
drop table map_s_w;
drop table phrase_count;
drop table phrase_examples;
drop table quellen;
drop table saetze;
drop table sprachen;
drop table status;
drop table tagset_ims_de;
drop table tagset_ims_en;
drop table tagset_lql_de;
drop table tagset_lql_en;
drop table tagsets;
drop table tokens;
```

O.2 Python-Datenbank-Schnittstelle

Das Python-Paket `DatabaseAPI` befindet sich im CVS Baum im Verzeichnis `align`.

O.2.1 __init__.py

```
#!/usr/bin/python

# Initialisierung des Pakets "DatabaseAPI"

__all__ = [                   # Module als DatabaseAPI.xxx ansprechbar machen
           "absatzdb.py",
           "atomdb.py",
           "cache.py",
           "config.py"
           "db.py",
           "dokumentdb.py",
           "korpusdb.py",
           "qclient.py",
           "querydb.py",
           "regiondb.py",
           "satzdb.py",
           "segmentdb.py",
]

# nicht oeffentlich:
#           "alignmentdb.py",
#           "hindex.py",
#           "secret.py",
```

O.2.2 RegionVonDB

`regiondb.py`

Die Grundklasse aller Datenbank-Regionen.

```
#!/usr/bin/python

# Modul DatabaseAPI.regiondb

from Region import region
import db
import sys
import types

defaultDbconn = None

def setDefaultDbconn(dbconn):
  global defaultDbconn
  defaultDbconn = dbconn

class RegionVonDb(region.Regionen):

  def __init__(self, id, dbconn):
      self.dbconn = dbconn
      self.subregionen = {}
      self.regionid = id
      self.idliste = self.loadids(id)

  def __repr__(self):   #ueberschreiben
      return "<%s id=%s>" %(self.__class__.__name__, self.regionid) #, self.subregionen)

  def __getstate__(self):
      return (self.idliste, self.regionid, self.subregionen)

  def __setstate__(self, state):
```

```
        self.dbconn = defaultDbconn
        self.idliste = state[0]
        self.regionid = state[1]
        #self.subregionen = {}
        self.subregionen = state[2]

    def setDbconn(self, dbconn):
        self.dbconn = dbconn
        for key in self.subregionen.keys():
            self.subregionen[key].setDbconn(dbconn)

    def loadids(self, id): #ueberschreiben
        return []

    def loaditem(self, key, pos): #ueberschreiben
        return ""

    def loadquery(self, query):
        #sys.stderr.write("RegionVonDb.loadquery("+‘query‘+")\n")
        ids = []
        result = self.dbconn.executelist(query)
        if result and result[0]:
            ids = map(lambda x: db.unknown2string(x[0]), result)
        return ids

    def __len__(self):
        #if type(self.idliste) != types.ListType:
        #    sys.stderr.write("Fehler: type(RegionVonDb.idliste) = "+‘type(self.idliste)‘+"\n")
        return len(self.idliste)

    def getItemAtIndex(self, key):
        if not self.subregionen.has_key(key):
            self.subregionen[key] = self.loaditem(self.idliste[key])
        return self.subregionen[key]


```

alignmentdb.py

Repräsentation einer alignten Region.

```
#!/usr/bin/python

# Modul DatabaseAPI.alignmentdb

import regiondb, db
from Region import alignment
import config
import sys
import types

class AlignedRegionVonDb(regiondb.RegionVonDb, alignment.AlignedRegion): #{

    def __init__(self, id, dbconn, sprache):
        self.sprache = sprache
        self.alregid = id
        regiondb.RegionVonDb.__init__(self, id, dbconn)

    def __repr__(self):
        return "<"+db.unknown2string(self.alregid)+" "+\
        self.__class__.__name__+" "+\
        db.unknown2string(self.sprache)+" "+\
```

```
22        ("dbconn=%s" %‘id(self.dbconn)‘)+">"
23
24    def alignedRegion(self):
25        sprache = None
26        for i in config.lang2id.values():
27            if i != self.sprache:
28                sprache = i
29        if sprache:
30            return self.__class__(self.alregid, self.dbconn, sprache)
31        else:
32            sys.stderr.write("AlignedRegionVonDb: alignte Sprache unklar.\n")
33
34    def __setstate__(self, state):
35        regiondbstate, sprache, alregid = state
36        regiondb.RegionVonDb.__setstate__(self, regiondbstate)
37        self.sprache = sprache
38        self.alregid = alregid
39
40    def __getstate__(self):
41        regiondbstate = regiondb.RegionVonDb.__getstate__(self)
42        return (regiondbstate, self.sprache, self.alregid)
43
44  #}
```

atomdb.py

Repräsentation eines Wortes.

```
1   #!/usr/bin/python
2
3   # Modul DatabaseAPI.atomdb
4
5   from Region import atom
6   import db, regiondb
7   import sys
8
9   #Problem: nimmt immer nur daserste Token!!!! TODO
10  class AtomVonDb(atom.Atom):
11
12    def __repr__(self):
13        return "<AtomVonDB " +‘( self.feld["Wort"],
14                          self.feld["Tag"],
15                          self.feld["Lemma"] )‘ + ">"
16
17    def __init__(self, wort, dbconn, id = 0):
18        # atom.Atom.__init__(self)   # Aufruf erfolgt in loadFromDb
19        self.dbconn = dbconn
20        self.loadFromDb(wort, id)
21
22    def __getstate__(self):
23        return self.id
24
25    def __setstate__(self, state):
26        self.dbconn = regiondb.defaultDbconn
27        self.loadFromDb("", state)
28
29    def setDbconn(self, dbconn):
30        self.dbconn = dbconn
31
32    def loadFromDb(self, wort, id):  # id ist String oder Int, aber kein Long
33        tag = "<none>"
34        lemma = "<unknown>"
```

```
      if id:
        where = "tokens.tokenid = %s" %id
      else:
        where = "tokens.name = '%s'" %wort
      results = self.dbconn.executelist("select distinct grundform.name, tokens.tagid, tokens.tagsetid, token
      if results:
          lemma = results[0][0]
          tagid = db.unknown2string(results[0][1])     # dieses spaeter???
          tagsetid = db.unknown2string(results[0][2])
          wort = results[0][3]
          self.id = db.unknown2string(results[0][4])
          results = self.dbconn.executelist("select name from tagsets where tagsetid = %s" %tagsetid)
          if results:
              tagset = db.unknown2string(results[0][0])
              results = self.dbconn.executelist("select name from %s where tagid = %s" %(tagset, tagid))
              tag = db.unknown2string(results[0][0])
      atom.Atom.__init__(self, wort, tag, lemma)

def test():
  dbconn = db.Db()
  for i in ("hat", "is", "pension", "Steuer", "er", "and"):
    atom = AtomVonDb(i, dbconn)
    print i, atom, 'atom', atom.getWoerter()
  for i in (3, 76, 129):
    atom = AtomVonDb("", dbconn, i)
    print i, atom, 'atom', atom.getWoerter()

if __name__ == '__main__':
  test()
```

satzdb.py

Repräsentation eines Satzes.

```
#!/usr/bin/python

# Modul DatabaseAPI.satzdb

from Region import satz
import regiondb, atomdb, db
import sys

class SatzVonDb(regiondb.RegionVonDb, satz.Satz):

  def __init__(self, id, dbconn):
      regiondb.RegionVonDb.__init__(self, id, dbconn)

  #def __repr__(self):   #ueberschreiben
  #  return "<Satz von Db>" # + 'self.subregionen' + ">"

  def loadids(self, id):
      #sys.stderr.write("SatzVonDb.loadids("+'id'+")\n")
      retval = self.loadquery("select tokenid from map_s_w where satzid = %s order by mapid" %id)
      #sys.stderr.write("idliste = " + 'retval' + "\n")
      return retval

  def loaditem(self, key):
      return atomdb.AtomVonDb("", self.dbconn, key)

def test():
    dbconn = db.Db()
    for i in (1, '23', 40, '190000', 397433, '400293'):
```

```
        satz = SatzVonDb(i, dbconn)
        print '(i, satz, satz.getText())'

if __name__ == '__main__':
    test()
```

segmentdb.py

Repräsentation eines Segments.

```
#!/usr/bin/python

from Region import segment
import satzdb, db, alignmentdb
import sys

class SegmentVonDb(alignmentdb.AlignedRegionVonDb, segment.Segment):

  def __init__(self, id, dbconn, sprache):
      alignmentdb.AlignedRegionVonDb.__init__(self, id, dbconn, sprache)

  def loadids(self, id):
      #sys.stderr.write("SegmentVonDb.loadids: " + '(id, self.sprache)' + "\n")
      return self.loadquery("select distinct m.satzid from map_s_w m, saetze s where m.segmentnr = %s and m.satz

  def loaditem(self, key):
      return satzdb.SatzVonDb(key, self.dbconn)

def test():
    dbconn = db.Db()
    for i in (1, 2, 150000, 155100, 167700, 179888, 190000):
        de = SegmentVonDb(i, dbconn, "1")
        en = de.alignedRegion()
        print '(i, de, en)', de.getText(), en.getText()

if __name__ == '__main__':
    test()
```

absatzdb.py

Repräsentation eines Absatzes.

```
#!/usr/bin/python
from Region import absatz
import segmentdb, db, alignmentdb

#bis jetzt ist ein Absatz das gleiche wie ein Dokument!

class AbsatzVonDb(alignmentdb.AlignedRegionVonDb, absatz.Absatz):

  def __init__(self, id, dbconn, sprache):
      alignmentdb.AlignedRegionVonDb.__init__(self, id, dbconn, sprache)

  def loadids(self, id):
      return self.loadquery("select distinct m.segmentnr from map_s_w m, saetze s where s.herkunftsid = %s and s

  def loaditem(self, key):
      return segmentdb.SegmentVonDb(key, self.dbconn, self.sprache)

def test():
    dbconn = db.Db()
```

```
    for i in (10,):
        absatz = AbsatzVonDb(i, dbconn, "1")
        print ‘absatz‘
        print absatz[1][0][0]
        print absatz[1][0]
        print absatz[1]
        print absatz.alignedRegion()[1]

if __name__ == '__main__':
    test()
```

dokumentdb.py

Repräsentation eines Dokuments.

```
#!/usr/bin/python
from Region import dokument
import absatzdb, db, alignmentdb

#bis jetzt ist ein Absatz das gleiche wie ein Dokument!

class DokumentVonDb(alignmentdb.AlignedRegionVonDb, dokument.Dokument):

  def __init__(self, id, dbconn, sprache):
      alignmentdb.AlignedRegionVonDb.__init__(self, id, dbconn, sprache)

  def loadids(self, id):
      return [id]

  def loaditem(self, key):
      return absatzdb.AbsatzVonDb(key, self.dbconn, self.sprache)

def test():
    dbconn = db.Db()
    for i in (1, 2):
        dokument = DokumentVonDb(i, dbconn, "1")
        print ‘dokument‘

if __name__ == '__main__':
    test()
```

korpusdb.py

Repräsentation eines Korpus.

```
#!/usr/bin/python
from Region import korpus
import dokumentdb, db, alignmentdb

#bis jetzt ist ein Absatz das gleiche wie ein Dokument!
class KorpusVonDb(alignmentdb.AlignedRegionVonDb, korpus.KorpusMitSprache):

  def __init__(self, dbconn, sprache):
      alignmentdb.AlignedRegionVonDb.__init__(self, 0, dbconn, sprache)

  def __repr__(self):   #ueberschreiben
      return "<Korpus von Db>"

  def loadids(self, id):
      ids = []
      result = self.dbconn.executelist("select herkunftsid from herkunft")
      if result and result[0]:
```

```
            ids = map(lambda x: x[0], result)
        return ids

    def loaditem(self, key):
        return dokumentdb.DokumentVonDb(key, self.dbconn, self.sprache)


#Falle! Sprachids sind "1" => deutsch, "2" => englisch
def test():
    dbconn = db.Db()
    for i in ("1", "2"):
        korpus = KorpusVonDb(dbconn, i)
        print korpus[0][0][0][0]

if __name__ == '__main__':
    test()
```

O.2.3 Wörterbücher

`wb.py`

Ein Wörterbuch, aus einer Datei oder der Datenbank.

```
#!/usr/bin/python

# Modul wb

from DatabaseAPI import cache
import zeitmessung
import string
import sys
import time
import types
# weitere imports nach den Klassendefinitionen

class Lemma2src(cache.Cached): #{

  def __init__(self, client, quellsprache, zielsprache):
    cache.Cached.__init__(self, client)
    self.quellsprache = quellsprache
    self.zielsprache  = zielsprache

  def loaditem(self, key):
    return []

  def saveitem(self, key, value):
    pass

#}

class Src2dst(cache.Cached): #{

  def __init__(self, client, quellsprache, zielsprache):
    cache.Cached.__init__(self, client)
    self.quellsprache = quellsprache
    self.zielsprache  = zielsprache

  def loaditem(self, key):
    return []

  def saveitem(self, key, value):
    pass
```

```
40
41  #}
42
43  class Woerterbuch(zeitmessung.Zeitmessung): #{
44
45      def __init__(self, quellsprache, zielsprache):
46          zeitmessung.Zeitmessung.__init__(self)
47          self.lemma2src = Lemma2src(self, quellsprache, zielsprache)
48          self.src2dst   = Src2dst(self, quellsprache, zielsprache)
49          self.quellsprache = quellsprache
50          self.zielsprache  = zielsprache
51
52      def __repr__(self):
53          return "<WB %s nach %s>" %(‘self.quellsprache‘,‘self.zielsprache‘)
54
55      def __getitem__(self, lemma):
56          self.p_begin("getitem")
57          srcliste = self.lemma2src[lemma]
58          retval=[]
59          for (pos, srcseg) in srcliste:
60              retval.append((pos, srcseg, self.src2dst[‘srcseg‘]))
61          self.p_end("getitem")
62          return retval
63
64      def has_key(self, lemma):
65          self.p_begin("has_key")
66          retval = self.lemma2src.has_key(lemma)
67          self.p_end("has_key")
68          return retval
69
70  #}
71
72  ##############################################################################
73  #
74  # VonDatei
75
76  class Lemma2srcVonDateien(Lemma2src): #{
77
78    def __init__(self, client, quellsprache, zielsprache):
79      Lemma2src.__init__(self, client, quellsprache, zielsprache)
80      if not isinstance(self, cache.FullCache):
81        sys.stderr.write("Warnung: Lemma2srcVonDateien basiert nicht auf FullCache\n")
82
83    def addSegmentpaar(self, seg_s, seg_d):
84      pos = 1                             # Bei jedem Lemma seg_s eintragen
85      for lemma in seg_s.getLemmata(): #{
86        if self.has_key(lemma):
87          srcliste = self[lemma]
88        else:
89          srcliste = []
90        srcliste.append((pos, seg_s))    # bei der Gelegenheit pos merken
91        self[lemma] = srcliste
92        pos = pos + 1
93      #}
94  #}
95
96  class Src2dstVonDateien(Src2dst): #{
97
98    def __init__(self, client, quellsprache, zielsprache):
99      Src2dst.__init__(self, client, quellsprache, zielsprache)
100     if not isinstance(self, cache.FullCache):
```

```
101       sys.stderr.write("Warnung: Src2dstVonDateien basiert nicht auf FullCache\n")
102
103     def addSegmentpaar(self, seg_s, seg_d):
104       srckey = ‘seg_s‘                    # Paar (seg_s, seg_d) eintragen
105       if self.has_key(srckey):
106         dstliste = self[srckey]
107       else:
108         dstliste = []
109       dstliste.append(seg_d)
110       self[srckey] = dstliste
111
112   #}
113
114   class WoerterbuchVonDateien(Woerterbuch): #{
115
116     def __init__(self, quellsprache, zielsprache): #{
117       Woerterbuch.__init__(self, quellsprache, zielsprache)
118       self.lemma2src = Lemma2srcVonDateien(self, quellsprache, zielsprache)
119       self.src2dst   = Src2dstVonDateien(self, quellsprache, zielsprache)
120       for wbdatei in config.wbdateien: #{
121         self.addWB(wbdatei)
122       #}
123     #}
124
125     def addWB(self, wbdatei): #{
126       self.p_begin("addWB")
127       if  wbdatei.has_key(self.quellsprache) \
128       and wbdatei.has_key(self.zielsprache): #{
129         src = dokument.DokumentVonDatei(
130               wbdatei[self.quellsprache]).getRegionenMit(segment.Segment)
131         dst = dokument.DokumentVonDatei(
132               wbdatei[self.zielsprache]).getRegionenMit(segment.Segment)
133         anzahl = len(src)
134         if anzahl > len(dst): #{
135           sys.stderr.write(’Woerterbuch ’ + ‘wbdatei‘ + ’ weniger Segmente ’ + \
136                            ’in der Sprache "’ + self.zielsprache + ’" als ’ + \
137                            ’in der Sprache "’ + self.quellsprache + ’".\n"’)
138           anzahl = len(dst)
139         #}
140         sys.stderr.write("Verarbeite " + ‘anzahl‘ + " Segmente:\n")
141         letzteAusgabe = 0
142         for i in range(0,anzahl): #{
143           if time.time() - letzteAusgabe >= 1.0:
144             sys.stderr.write(‘100*i/anzahl‘ + "% erledigt\r")
145             letzteAusgabe = time.time()
146           seg_s = src[i]
147           seg_d = dst[i]
148           self.lemma2src.addSegmentpaar(seg_s, seg_d)
149           self.src2dst.addSegmentpaar(seg_s, seg_d)
150         #}
151         sys.stderr.write("100% erledigt\n")
152         self.p_end("addWB")
153         return 1
154       #}
155       self.p_end("addWB")
156       return 0
157     #}
158
159     def debug(self):
160       print "### lemma2src ###"
161       for i in self.lemma2src.keys():
```

```
      print ("[" + `i` + "]: " + `self.lemma2src[i]`)
    print "### src2dst ###"
    for i in self.src2dst.keys():
      print ("[" + `i` + "]: " + `self.src2dst[i]`)
#}

##############################################################################
#
# VonDb

class WoerterbuchVonDB(Woerterbuch): #{

  def __init__(self, quellsprache, zielsprache): #{
    Woerterbuch.__init__(self, quellsprache, zielsprache)
    self.dbq = qclient.Query()
    if not self.dbq.ok():
        sys.stderr.write("Query-Server nicht gefunden!\n")

  def close(self):
    self.dbq.close()

  def has_key(self, key):
    self.p_begin("has_key")
    self.dbq.setwbkorpus()
    q = self.dbq.segment(match=key, lang=config.lang2id[self.quellsprache])
    if q:
        retval = len(q)
    else:
        retval = 0
    self.p_end("has_key")
    return retval

  def __getitem__(self, woerter):
    self.p_begin("getitem")
    liste = []
    self.dbq.setwbkorpus()
    q = self.dbq.segment(match=woerter, lang=config.lang2id[self.quellsprache])
    if q:
        for seg in q:
            dst = seg.alignedRegion()
            liste.append((-1, seg, [dst]))
    else:
        f = open("wort_nicht_im_wb", "a")
        f.write("%s\n" %woerter)
        f.close()
    self.p_end("getitem")
    return liste

#}

class LemmatabuchVonDB(Woerterbuch): #{

    def __init__(self, quellsprache, zielsprache, qdb_type="wb"): #{
        Woerterbuch.__init__(self, quellsprache, zielsprache)
        self.dbq = qclient.Query()
        self.dbq.setkorpus(qdb_type)
        if not self.dbq.ok():
            sys.stderr.write("Query-Server nicht gefunden!\n")
        self.cache = cache.DefaultCache(self)

    def close(self):
```

```
223         self.dbq.close()
224 
225     def has_key(self, key):
226         self.p_begin("has_key")
227         if "<unknown>" in key:              # 2do
228             self.p_end("has_key")
229             return 0
230         lstr = string.join(key)
231         ckey = (0, lstr)
232         if self.cache.has_key(ckey):
233             retval = self.cache[ckey]
234             self.p_end("has_key")
235             return retval
236         for i in ",:;=.!?@":   # Satzzeichen, @card@ , etc.
237             if i in lstr:
238                 self.cache[ckey] = 0
239                 self.p_end("has_key")
240                 return 0
241         ckey2 = (3, lstr)
242         if self.cache.has_key(ckey2):
243             q = self.cache[ckey2]
244         else:
245             self.p_begin("has_key_query")
246             q = self.dbq.segment(match_lf = lstr,
247                                  strict   = "true",
248                                  max      = "1",
249                                  lang     = config.lang2id[self.quellsprache])
250             self.p_end("has_key_query")
251             self.cache[ckey2] = q
252         if q:
253             retval = len(q)
254         else:
255             retval = 0
256         self.cache[ckey] = retval
257         self.p_end("has_key")
258 #        print retval
259         return retval
260 
261     def __getitem__(self, lemmata):
262         self.p_begin("getitem")
263         liste = []
264         if "<unknown>" in lemmata:              # 2do
265             self.p_end("getitem")
266             return liste
267         lstr = string.join(lemmata)
268         ckey = (1, lstr)
269         if self.cache.has_key(ckey):
270             retval = self.cache[ckey]
271             self.p_end("getitem")
272             return retval
273         for i in ",:;=.!?@":   # Satzzeichen, @card@ , etc.
274             if i in lstr:
275                 self.cache[ckey] = liste
276                 self.p_end("getitem")
277                 return liste
278         ckey2 = (2, lstr)
279         if self.cache.has_key(ckey2):
280             q = self.cache[ckey2]
281         else:
282             self.p_begin("getitem_query")
283             q = self.dbq.segment(match_lf = string.join(lemmata),
```

```
                                strict   = "true",
                                max      = "100",
                                lang     = config.lang2id[self.quellsprache])
            self.p_end("getitem_query")
            self.cache[ckey2] = q
        if q:
            for seg in q:
                dst = seg.alignedRegion()
                liste.append((-1, seg, [dst]))
        else:
            f = open("lemma_nicht_im_wb", "a")
            f.write("%s\n" %lemmata)
            f.close()
        self.cache[ckey] = liste
        self.p_end("getitem")
        return liste


def test():
  regiondb.setDefaultDbconn(db.Db())
  wb_deen = WoerterbuchVonDB("de", "en")
  wb_ende = WoerterbuchVonDB("en", "de")
  for (wb, liste) in [(wb_deen, ['Apfel', 'Baum', 'Apfelbaum']),
                      (wb_ende, ['apple', 'tree', 'apple tree'])]:
    print 'wb'
    for i in liste:
      print 'i'
      for (pos, src, dstlist) in wb[i]:
        print '\tde:', src.getText()
        for j in dstlist:
          print '\ten:', j.getText()

from Region      import dokument, segment
from DatabaseAPI import qclient, regiondb, db, config

if __name__ == '__main__':
  test()
```

filterwb.py

Ein Wörterbuch, mit dem man nach Tags die Relevanz eines Wortes ermitteln kann.

```
#!/usr/bin/python

# Modul filterwb

import zeitmessung
import base64
import pickle
import string
import sys
import types
# weitere imports nach den Klassendefinitionen

class Relevant: #{

    def __init__(self, sprache):
        self.sprache = sprache

    def relevant(self, region):
        return 1
```

```
20
21      def p_getSlices(self, regionen, atome):
22          return range(len(atome))
23
24      def __call__(self, regionen):
25          atome = regionen.getRegionenMit(atom.Atom)
26          liste = self.p_getSlices(regionen, atome)
27  #        print liste
28          relliste = []
29          for slice in liste:
30              if self.relevant(atome[slice]):
31                  relliste.append(slice)
32                  #print "Slice",‘slice‘,"OK"
33              else:
34                  #print "Slice",‘slice‘,"verworfen"
35                  pass
36          return (atome, relliste)
37
38  #}
39
40  #
41  # Python 2.1 workaround for slice objects
42  #
43  #   "Slice objects are now comparable but not hashable; this prevents
44  #   dict[:] from being accepted but meaningless."
45  #   (Python 2.1 Release Notes)
46  #
47
48  import UserDict
49
50  class Dictionary(UserDict.UserDict): #{
51
52      def __init__(self):
53          UserDict.UserDict.__init__(self)
54          self.okey = {}
55
56      def __getitem__(self, key):
57          if type(key) == types.SliceType:
58              mkey = (’Slice’, key.start, key.stop, key.step)
59          else:
60              mkey = key
61          return UserDict.UserDict.__getitem__(self, mkey)
62
63      def keys(self):
64          mkeys = UserDict.UserDict.keys(self)[:]   # copy list of keys
65          for index in range(len(mkeys)):
66              try:
67                  okey = self.okey[mkeys[index]]
68                  mkeys[index] = okey
69              except KeyError:
70                  pass
71          return mkeys
72
73      def __setitem__(self, key, value):
74          if type(key) == types.SliceType:
75              mkey = (’Slice’, key.start, key.stop, key.step)
76          else:
77              mkey = key
78          self.okey[mkey] = key
79          UserDict.UserDict.__setitem__(self, mkey, value)
80
```

```
    def has_key(self, key):
        if type(key) == types.SliceType:
            mkey = ('Slice', key.start, key.stop, key.step)
        else:
            mkey = key
        return UserDict.UserDict.has_key(self, mkey)
#}


class WoerterbuchFilter(zeitmessung.Zeitmessung): #{

    def __init__(self, quellsprache, zielsprache, relevant, qdb_type="wb"):
        zeitmessung.Zeitmessung.__init__(self)
        self.wb = wb.LemmatabuchVonDB(quellsprache, zielsprache, qdb_type)
        self.quellsprache = quellsprache
        self.zielsprache  = zielsprache
        self.relevant = relevant
        self.cache = cache.DefaultCache(self)

    def close(self):
        self.wb.close()

    def __repr__(self):
        return "<WoerterbuchFilter %s zu %s>" %(self.quellsprache,
                                                self.zielsprache)

    def __getitem__(self, regionen):
        self.p_begin("getitem")
 #       print "getitem!", 'regionen'
        if type(regionen) == types.TupleType:  # neues Format?
            format_neu = 1
            markiert = regionen[0]             # Indizes der markierten Woerter
            regionen = regionen[1]
        else:
            format_neu = 0
        if self.cache.has_key(id(regionen)):   # 2do: (w,t,l) benutzen?
            retval = self.cache[id(regionen)]
            self.p_end("getitem")
            return retval
        retval = Dictionary()
        self.p_begin("relevant")
        atome, slices = self.relevant(regionen)
        if not format_neu:                     # altes Format?
            markiert = range(len(atome))       # -> alles markiert
        self.p_end("relevant")
        self.p_begin("for bereich in slices")
        vt = 'o'
        for bereich in slices:
            if type(bereich) == types.IntType:
                if vt == 'o':
                    self.p_begin("type(bereich) == types.IntType")
                    vt = 'i'
                elif vt == 's':
                    self.p_end("type(bereich) != types.IntType")
                    self.p_begin("type(bereich) == types.IntType")
                    vt = 'i'
                index   = bereich
                if not index in markiert:
                    continue
                self.p_begin("atome[index]")
                einAtom = atome[index]
```

```
            self.p_end("atome[index]")
            if not retval.has_key(index):
                self.p_begin("wb.has_key")
                wbhaskey = self.wb.has_key(einAtom.getLemmata())
                self.p_end("wb.has_key")
                if wbhaskey:
                    self.p_begin("eintragsliste bestimmen")
                    eintragsliste = self.wb[einAtom.getLemmata()]
                    self.p_end("eintragsliste bestimmen")
                    segmente = {}
                    self.p_begin("eintragsliste durchgehen")
                    for (pos, srcseg, dstsegliste) in eintragsliste:
                        for dstseg in dstsegliste:
                            if format_neu:
                                segmente[srcseg.regionid] = (srcseg, dstseg)
                            else:
                                segmente[dstseg.getText()] = dstseg  # unique machen
                    self.p_end("eintragsliste durchgehen")
                    if format_neu:
                        retval[index] = segmente.values()
                    else:
                        self.p_begin("Absatz bauen")
                        retval[index] = absatz.Absatz(segmente.values())
                        self.p_end("Absatz bauen")
                else:
                    if format_neu:
                        retval[index] = []
                    else:
                        retval[index] = absatz.Absatz([]) # leerer Absatz
        else:
            if vt == 'o':
                self.p_begin("type(bereich) != types.IntType")
                vt = 's'
            elif vt == 'i':
                self.p_end("type(bereich) == types.IntType")
                self.p_begin("type(bereich) != types.IntType")
                vt = 's'
            etwas_markiert = 0
            for index in range(bereich.start, bereich.stop):
                if index in markiert:
                    etwas_markiert = 1
                    break
            if not etwas_markiert:
                continue
            atomliste = atome[bereich]
            if not retval.has_key(bereich):
                if self.wb.has_key(atomliste.getLemmata()):
                    eintragsliste = self.wb[atomliste.getLemmata()]
                    segmente = {}
                    self.p_begin("eintragsliste durchgehen")
                    for (pos, srcseg, dstsegliste) in eintragsliste:
                        for dstseg in dstsegliste:
                            if format_neu:
                                segmente[srcseg.regionid] = (srcseg, dstseg)
                            else:
                                segmente[dstseg.getText()] = dstseg  # unique machen
                    self.p_end("eintragsliste durchgehen")
                    if format_neu:
                        retval[bereich] = segmente.values()
                    else:
                        self.p_begin("Absatz bauen")
```

```
                            retval[bereich] = absatz.Absatz(segmente.values())
                            self.p_end("Absatz bauen")
                    else:
                        if format_neu:
                            retval[bereich] = []
                        else:
                            retval[bereich] = absatz.Absatz([]) # leerer Absatz
        if vt == 's':
            self.p_end("type(bereich) != types.IntType")
        elif vt == 'i':
            self.p_end("type(bereich) == types.IntType")
        self.p_end("for bereich in slices")
#        print retval
        self.cache[id(regionen)] = retval
        self.p_end("getitem")
        return retval
        # Format: { 'atom1' : [seg11, seg12, ..., seg1n], ([] als Absatz)
        #           'atom4' : ... } # nur relevante atomi
        # wobei atomi = unique atoms in regionen
        #       segij = WB-Eintrag zu lemma(atomi)

    def has_key(self, regionen):
        return 1
#}

class RelevantMax(Relevant): #{

    def __init__(self, max, sprache):
        Relevant.__init__(self, sprache)
        self.max = max

    def p_getSlices(self, regionen, atome):
        liste = Relevant.p_getSlices(self, regionen, atome)
        for startIndex in range(len(atome)):
            ende = startIndex+self.max
            if ende >= len(atome):
                ende = len(atome)-1
            for endIndex in range(startIndex+2, ende+1):
                liste.append(slice(startIndex, endIndex))
        return liste

#}

class RelevantNachTagP(RelevantMax): #{

    def __init__(self, sprache, schwelle = 0, max = 4):
        RelevantMax.__init__(self, max, sprache)
        self.positiv = tags[config.lang2id[sprache]]
        self.schwelle = schwelle

    def relevant(self, region):
        tags = region.getTags()
        rel = 0
        for tag in tags:
            if self.positiv.has_key(tag):
                if self.positiv[tag] == -2: #alles, was mit -2 markiert ist, raus
                     return 0.0
                rel = rel + self.positiv[tag]
#        print rel
        #output = open("tags_all_"+self.sprache+".out", "a")
        #output.write(string.join(tags, '\t') + '\n')
```

```
        #output.close()
        return rel > (self.schwelle * len(tags))

#}

class RelevantWegenNomen(RelevantMax): #{

    def __init__(self, sprache, schwelle = 0, max = 4):
        RelevantMax.__init__(self, max, sprache)
        self.schwelle = schwelle

    def relevant(self, region):
        tags = region.getTags()
        rel = 0
        for tag in tags:
            if tag[0] != 'N':  #alles, was nicht mit N anfaengt, raus
                 return 0.0
            rel = rel + 2
        return rel > (self.schwelle * len(tags))


class RelevantInseln(RelevantNachTagP): #{

    def __init__(self, sprache, schwelle = 0):
        self.inseltags = self.p_loadTags(sprache, config.tags)
        RelevantNachTagP.__init__(self, sprache, schwelle, max(self.inseltags.values()))

    def p_getSlices(self, regionen, atome):
        liste = RelevantNachTagP.p_getSlices(self, regionen, atome)
        indizes = liste[:]
        for sl in indizes:
            if not self.inseltags.has_key(string.join(atome[sl].getTags())):
                liste.remove(sl)
        return liste

    def p_loadTags(self, sprache, dir):
        tags = {}
        tagfile = open(dir + "/tags_" + sprache + ".txt", "r")
        for line in tagfile.readlines():
            tagline = string.strip(line)
            length = len(string.split(tagline))
            tags[tagline] = length
        return tags

#}


class WoerterbuchFilterNachTagPOhneInseln(WoerterbuchFilter): #{

    def __repr__(self):
        return WoerterbuchFilter.__repr__(self)[:-1] +\
               ", der nur Atome mit positivem Tag durchlaesst>"

    def __init__(self, quellsprache, zielsprache, schwelle=0, qdb_type="wb"):
        relevant = RelevantNachTagP(quellsprache, schwelle)
        WoerterbuchFilter.__init__(self, quellsprache, zielsprache,
                                   relevant, qdb_type)
#}

class WoerterbuchFilterNachTagP(WoerterbuchFilter): #{
```

```
    def __repr__(self):
        return WoerterbuchFilter.__repr__(self)[:-1] +\
               ", der nur Atome mit positivem Tag durchlaesst>"

    def __init__(self, quellsprache, zielsprache, schwelle=0, qdb_type="wb"):
        relevant = RelevantInseln(quellsprache, schwelle)
        WoerterbuchFilter.__init__(self, quellsprache, zielsprache,
                                   relevant, qdb_type)
#}


class RelevantNachTagN(RelevantMax): #{

    def __init__(self, sprache, schwelle = 0, max = 4):
        RelevantMax.__init__(self, max, sprache)
        self.negativ = tags[config.lang2id[sprache]]
        self.schwelle = schwelle

    def relevant(self, atom):
        tags = atom.getTags()[0]            # ??? atom ist beliebige region !!!
        rel = 0
        for tag in tags:
            if self.negativ.has_key(tag):
                rel = rel + self.negativ[tag]
        return rel > self.schwelle
#}

class WoerterbuchFilterNachTagN(WoerterbuchFilter):

    def __repr__(self):
        return WoerterbuchFilter.__repr__(self)[:-1] +\
               ", der nur Atome mit negativem Tag ausschliesst>"

    def __init__(self, quellsprache, zielsprache, schwelle=0, qdb_type="wb"):
        relevant = RelevantNachTagN(quellsprache, schwelle)
        WoerterbuchFilter.__init__(self, quellsprache, zielsprache,
                                   relevant, qdb_type)


def test():
    regiondb.setDefaultDbconn(db.Db())
    wbf_p = WoerterbuchFilterNachTagP("de", "en")
    wbf_n = WoerterbuchFilterNachTagN("de", "en")
    #reg1 = segmentdb.SegmentVonDb(160000, db.Db(), "1")
    import tagger
    reg1 = tagger.tagline("Wir müssen jemanden in Kenntnis setzen.", "de")
    reg2 = tagger.tagline("Er kann nur mit Mühe und Not mit der Schreibmaschine schreiben.", "de")
    for (wbf, reg) in [(wbf_p, reg1), (wbf_p, reg2),
                       (wbf_n, reg1), (wbf_n, reg2)]:
        print "\n", `wbf`, "\n"
        print reg.getText()
        print reg.getTags()
        d = wbf[reg]
        a = reg.getRegionenMit(atom.Atom)
        #print "a:", `a`
        #print "d:", `d`
        for key in d.keys():
            #print 'Slice:', `key`
            print '\tde:', a[key].getText()
            print '\ten:', d[key].getText()
    regiondb.defaultDbconn.close()
```

```
386
387
388 from DatabaseAPI import  config, regiondb, db, atomdb, segmentdb, cache
389 from Region      import  atom, region, absatz
390 import wb
391
392 tags = { config.lang2id["de"] : { # vorlaeufige Klassifizierung von Philip
393                                   '$(': -2,
394                                   '$,': -2,
395                                   '$.': -2,
396                                   'ADJA': +1,
397                                   'ADJD': +1,
398                                   'ADV': 0,
399                                   'APPO': 0,
400                                   'APPR': 0,
401                                   'APPRART': 0,
402                                   'APZR': 0,
403                                   'ART': -2,
404                                   'CARD': +1,
405                                   'FM': -2,
406                                   'ITJ': 0,
407                                   'KOKOM': 0,
408                                   'KON': 0,
409                                   'KOUI': -1,
410                                   'KOUS': 0,
411                                   'NE': +2,
412                                   'NN': +2,
413                                   'PAV': -1,
414                                   'PDAT': -1,
415                                   'PDS': -1,
416                                   'PIAT': -1,
417                                   'PIDAT': -1,
418                                   'PIS': -1,
419                                   'PPER': -1,
420                                   'PPOSAT': -1,
421                                   'PPOSS': -1,
422                                   'PRELAT': -1,
423                                   'PRELS': -1,
424                                   'PRF': -1,
425                                   'PTKA': -1,
426                                   'PTKANT': -1,
427                                   'PTKNEG': 0,
428                                   'PTKVZ': +2,
429                                   'PTKZU': -1,
430                                   'PTKZV': 0,
431                                   'PWAT': -1,
432                                   'PWAV': -1,
433                                   'PWS': -1,
434                                   'SATZ-P': -2,
435                                   'TRUNC': -2,
436                                   'VAFIN': +2,
437                                   'VAIMP': +2,
438                                   'VAINF': +2,
439                                   'VAPP': +2,
440                                   'VMFIN': +2,
441                                   'VMINF': +2,
442                                   'VMPP': +2,
443                                   'VVFIN': +2,
444                                   'VVIMP': +2,
445                                   'VVINF': +2,
446                                   'VVIZU': +2,
```

```
                                'VVPP': +2,
                                'XY': -2,
                                #           LQL - Relevantliste von Arno
                                'adj': +1,
                                'adv': 0,
                                'art': -2,
                                'imp': -2,
                                'itj': -2,
                                'kon': -2,
                                'nn':  +2,
                                'prep':  -1,
                                'pron': -1,
                                'v':  +2,
                                'va': +2,
                                'vi': +2,
                                'vt': +2,
                              },
        config.lang2id["en"] : { # vorlaeufige Klassifizierung von Philip
                                '"': -2,
                                '#': -2,
                                '$': -2,
                                "'": -2,
                                '(': -2,
                                ')': -2,
                                ',': -2,
                                '.': -2,
                                ':': -2,
                                'CC': 0,
                                'CD': +1,
                                'DT': -2,
                                'EX': -2,
                                'FW': -2,
                                'IN': -1,
                                'JJ': +1,
                                'JJR': +1,
                                'JJS': +1,
                                'LS': -2,
                                'MD': +2,
                                'NN': +2,
                                'NNP': +2,
                                'NNPS': +2,
                                'NNS': +2,
                                'NP': +2,
                                'NPS': +2,
                                'PDT': -2,
                                'PP': -1,
                                'PP$': -1,
                                'PRP': -1,
                                'RB': -1,
                                'RBR': -1,
                                'RBS': -1,
                                'RP': -1,
                                'SATZ-P': -2,
                                'SYM': -2,
                                'TO': -2,
                                'UH': 0,
                                'VB': +2,
                                'VBD': +2,
                                'VBG': +2,
                                'VBN': +2,
                                'VBP': +2,
```

```
                                    'VBZ': +2,
                                    'WDT': -1,
                                    'WP': -1,
                                    'WP$': -1,
                                    'WRB': -1,
                                    #           LQL - Relevantliste von Arno
                                    'adj': +1,
                                    'adv': 0,
                                    'art': -2,
                                    'imp': -2,
                                    'itj': -2,
                                    'kon': -2,
                                    'nn':  +2,
                                    'prep':  -1,
                                    'pron': -1,
                                    'v':  +2,
                                    'va': +2,
                                    'vi': +2,
                                    'vt': +2,
                                },
        }


if __name__ == '__main__':
    test()

```

O.2.4 Datenbankserver und Hilfsklassen

`querydb.py`

Der Datenbankserver.

```
#!/usr/bin/python

from Region import region, absatz, atom, satz
import db, atomdb, satzdb, segmentdb, absatzdb, regiondb
import dokumentdb, cache, secret, config       # in current package
import filterwb, tagger                        # top-level module
import zeitmessung

import base64, getopt, pickle, popen2
import zlib
import os
import string, sys, thread, time, types
import SocketServer
from socket import *
import select
import MySQLdb

debug_level = 0        # Wert kann von Query.__init__ veraendert werden

todo = [
    "nicht nur q_obj cachen",
    "where und from Klauseln systematisch erzeugen",
    "LIMIT bei max verwenden",
    "cache bei update: pruefen",
    "join benutzen, z.B. " + """
         select distinct map_s_w.satzid
         from tokens left join map_s_w using ( tokenid )
         where tokens.name = "werden";
    """ + "deutlich schneller, Achtung: jedoch" + """
```

```
         select distinct t.tokenid
         from tokens t left join grundformen g using (grundformid)
         where g.name = 'Kenntnis';
    """ + "deutlich langsamer",
]

ERROR = "error!"

class Command:
    pass

class CommandDb(Command):

    def __init__(self, dbconndic):
        self.dbconn = dbconndic

class QueryBase(CommandDb):         # nur Basis von Query.+, nicht von Query

    def __init__(self, dbconn, line, usekorpora, args):
        CommandDb.__init__(self, dbconn)
        self.usage = "bad command!"
        self.korpora = usekorpora
        args.append("max")
        self.params = self.checkLine(line, args)
        self.query_type = ""
        self.public_queries = ["wort", "grundform", "tag"]
        self.public_selects = ["text"]
        self.public_wheres  = ["name", "lang", "lemma", "wort", "parse"]
        if not self.params:
            raise ERROR, self.usage

    def close(self):
        pass

    def __repr__(self):
        params = []
        for key in self.params.keys():
             params.append((key, self.params[key]))
        params.sort()                                   # feste Reihenfolge
        paramrep = 'params'
        if string.find(paramrep, "%") > -1:
            paramrep = string.replace(paramrep, "%", "%%")
        return ("<Query %s %s %s" %(paramrep,'self.korpora',
                                    'self.query_type')) + " %s>"

    def checkLine(self, line, args):
        listenkeys = [ "match", "match_l", "match_f", "match_lf",
                       "lemmata", "lem_inv",
                       "tags",
                     ]
        params = {}
        for item in string.split(line, ","):
            split = string.split(item, "=")
            if len(split) != 2 or not split[0] in args:
                raise ERROR, self.usage
            key   = split[0]
            value = split[1]
            # bei match muss es eine Liste sein,
            # sind aber nicht immer spaces
            if key in listenkeys:
                nvalue = []
```

```
91                  for wort in string.split(value):
92                      nvalue.append(string.replace(wort, "'", "''"))  # SQL
93                  value = nvalue
94              else:
95                  value = string.replace(value, "'", "''")  # SQL
96              params[key] = value
97          if params.has_key("strict"):
98              temp = string.lower(params["strict"])
99              if temp in ["true", "yes", "y", "1"]:
100                 params["strict"] = 1
101             else:
102                 del params["strict"]
103         if not params.has_key("strict"):
104             for key in listenkeys:
105                 if params.has_key(key):
106                     params[key].sort()          # feste Reihenfolge fuer cache
107         return params
108
109     def query(self, restricted, select):
110         if restricted:
111             access_denied = 0
112             if not self.query_type in self.public_queries:
113                 access_denied = 1
114             for i in self.params.keys():
115                 if i not in self.public_wheres:
116                     access_denied = 1
117                 if not select in self.public_selects:
118                     access_denied = 1
119             if access_denied:
120                 raise ERROR, "access denied!"
121         list = self.real_query()
122         max = len(list)
123         if self.params.has_key("max"):
124             max = int(self.params["max"])
125         return list, max
126
127     def real_query(self):
128         return []
129
130     def querydb(self, query):
131         debug("Query = %s\n" %`query`, 9)
132         list = self.dbconn["query"].executelist(query)
133         if list:
134             return self.loadids(list)
135         return []
136
137     def p_intersect(self, list1, list2):  # changes order of list1 and list2
138         list3 = []
139         list1.sort()
140         list2.sort()
141         debug(`self`+".p_intersect"+`(list1, list2)`+"\n", 25)
142         index1 = 0
143         index2 = 0
144         try:
145             while 1:
146                 while list1[index1] < list2[index2]:
147                     index1 = index1 + 1
148                 while list2[index2] < list1[index1]:
149                     index2 = index2 + 1
150                 if list1[index1] == list2[index2]:
151                     list3.append(list1[index1])
```

```
                    index1 = index1 + 1
                    index2 = index2 + 1
        except IndexError:
            pass
        debug(" -> "+'list3'+"\n", 25)
        return list3

    def loadids(self, idlist):
        return map(lambda x: db.unknown2string(x[0]), idlist)

class QueryWort(QueryBase):

    def __init__(self, dbconn, line, usekorpora, type = "wort"):
        QueryBase.__init__(self, dbconn, line, usekorpora, ["name"])
        self.dbquery = "select distinct t.tokenid from tokens t"
        if type == "wort":
            self.dbquery = self.dbquery + " where t.name ="
            self.query_type = "wort"
        elif type == "grundform":
            self.dbquery = self.dbquery + ", grundformen g where t.grundformid = g.grundformid and g.name ="
            self.query_type = "grundform"

    def __repr__(self):
        return QueryBase.__repr__(self) %self.dbquery

    def real_query(self):
        list = self.querydb(self.dbquery + " '%s' " %self.params["name"])
        if self.params.has_key("max"):
            list = list[:int(self.params["max"])]
        atomlist = []
        for id in list:
            atomlist.append(atomdb.AtomVonDb("", self.dbconn["regionen"], id))
        return atomlist


class WoerterbuchfilterNachBedarf:

    def __init__(self, qs, zs, fs, qdb_type_list):
        self.param = {}
        for qdb_type in qdb_type_list:
            self.param[(qs, qdb_type)] = (zs, fs)
            self.param[(zs, qdb_type)] = (qs, fs)
        self.cache = cache.DefaultCache(self)

    def close(self):
        for key in self.cache.keys():
            self.cache[key].close()

    def __getitem__(self, key):
        p = self.param[key]
        ckey = (key, p)
        if self.cache.has_key(ckey):
            return self.cache[ckey]
        retval = filterwb.WoerterbuchFilterNachTagPOhneInseln(key[0], p[0], p[1], key[1])
        self.cache[ckey] = retval
        return retval

    def has_key(self, key):
        return self.param.has_key(key)

    def keys(self):
```

```
213         return self.param.keys()
214
215
216 class QueryPhrase(QueryBase):
217
218     def __init__(self, dbconn, line, usekorpora, args):
219         QueryBase.__init__(self, dbconn, line, usekorpora, args)
220         if self.params.has_key("_fs"):
221             fs = float(self.params["_fs"])
222         else:
223             fs = 0.45
224         self.filter = WoerterbuchfilterNachBedarf("de", "en", fs, ["erkannt", "wb"])
225         self.stopwords = [
226             "not", "hat", "nicht", "as", "werden", "had", "with", "sich",
227             "dem", "das", "eine", "has", "by", "was", "mit", "will",
228             "is", "auf", "des", "im", "on", "an", "zu", "für",
229             "be", "von", "den", "that", "for", "a", "und", "and",
230             "die", "to", "der", "of", "in", "the",
231         ]
232         self.stoplemmata = [
233             "es", "from", "he", "this", "say", "bei", "nicht", "not",
234             "will", "at", "as", "with", "nach", "by", "mit", "auf",
235             "zu", "on", "an", "für", "im", "von", "haben", "that",
236             "werden", "for", "have", "sein", "und", "a", "and", "ein",
237             "to", "be", "of", "in", "the",
238         ]
239         self.langids = [-1]  # von Querys mit alignten Regionen neu zu belegen
240
241     def close(self):
242         self.filter.close()
243
244     def __repr__(self):
245         return QueryBase.__repr__(self) %"Phrase"
246
247     def p_froms(self, liste):
248         dic = {
249                'g'      : "grundformen",
250                'm'      : "map_s_w",
251                'ph_c'   : "phrase_count",
252                'ph_e'   : "phrase_examples",
253                's'      : "saetze",
254                't'      : "tokens",
255                'tag_de' : "tagset_ims_de",
256                'tag_en' : "tagset_ims_en",
257         }
258         if not liste:
259             raise ValueError, "keine Tabelle fuer from angegeben"
260         erledigt = []
261         s = ""
262         for i in liste:
263             if not i in erledigt:
264                 s = s + dic[i] + " " + i + ", "
265                 erledigt.append(i)
266         return "from " + s[:-2] + " "
267
268     def p_distinct(self, idliste): # MySQL distict seltsam bei group und having
269         idliste.sort()
270         isfirst = 1
271         last = 0
272         retval = []
273         for value in idliste:
```

```
            if isfirst or value != last:
                retval.append(value)
            last = value
            isfrist = 0
        return retval

    def p_query(self, column):
        # Achtung: nicht jede column - params Kombination funktioniert !
        froms = []
        if column[:2] == "s." or self.korpora != "":
            froms.append('s')
        list = []
        if self.params.has_key("id"):
            list = [self.params["id"]]
            # 2do: parse und andere params: behandeln oder usage
        elif self.params.has_key("satzid"):
            froms.append('m')
            list = self.querydb(
                ("select distinct %s " %column) +\
                self.p_froms(froms) +\
                ("where m.satzid = %s " %self.params["satzid"]) +\
                ("%s" %self.korpora)
            )
            # 2do: parse und andere params: behandeln oder usage
        elif self.params.has_key("wort"):
            froms.append('m')
            froms.append('t')
            list = self.querydb(
                ("select distinct %s " %column) +\
                self.p_froms(froms) +\
                ("where t.name = '%s' " %self.params["wort"]) +\
                ("and t.tokenid = m.tokenid ") +\
                ("%s" %self.korpora)
            )
        elif self.params.has_key("lemma"):
            froms.append('m')
            froms.append('t')
            froms.append('g')
            if self.params.has_key("parse"):
                froms.append('s')
                ew =  "and m.satzid = s.satzid " %s
                pval = self.params["parse"]
                if pval == "NULL":
                    ew = ew + "and s.parsequalitaet IS NULL "
                elif pval[:2] == "GT":
                    ew = ew + "and s.parsequalitaet > %s " %pval[2:]
                else:
                    ew = ew + "and s.parsequalitaet = %s " %pval
            else:
                ew = ""
            list = self.querydb(
                ("select distinct %s " %column) +\
                self.p_froms(froms) +\
                ("where g.name = '%s' " %self.params["lemma"]) +\
                ("and t.tokenid = m.tokenid ") +\
                ew +\
                ("and g.grundformid = t.grundformid %s" %self.korpora)
            )
        elif self.params.has_key("match"):
            phrasewords = self.params["match"][:]
            for word in phrasewords[:]:
```

```
            if word in self.stopwords:
                phrasewords.remove(word)
        if not phrasewords:
            for stopword in self.stopwords:
                if stopword in self.params["match"]:
                    phrasewords = [stopword]
                    break
        if not phrasewords:
            raise ERROR, "match ohne Wert!"
        phraselen = len(phrasewords)
        phrase= string.join(phrasewords)
        phrasewds = "('"+string.join(phrasewords,"','")+"')"
        froms.append('m')
        froms.append('t')
        if self.params.has_key("parse"):
            froms.append('s')
            ew = "and s.satzid = m.satzid "
            pval = self.params["parse"]
            if pval == "NULL":
                ew = ew + "and s.parsequalitaet IS NULL "
            elif pval[:2] == "GT":
                ew = ew + "and s.parsequalitaet > %s " %pval[2:]
            else:
                ew = ew + "and s.parsequalitaet = %s " %pval
        else:
            ew = ""
        list = self.querydb(
            ("select distinct %s " %column) +\
            self.p_froms(froms) +\
            ("where t.name IN %s " %phrasewds) +\
            ("and t.tokenid = m.tokenid ") +\
            ("%s " %self.korpora)  +\
            ew +\
            ("group by m.satzid ") +\
            ("having count(m.tokenid) >= %s" %phraselen)
        )
    elif self.params.has_key("match_f"):
        phrasewords = self.params["match_f"][:]   # list
        wlen = len(phrasewords)
        wkey = [1, column, self.korpora] # muss sich von lkey unterscheiden
        if self.params.has_key("parse"):
            wkey.append(self.params["parse"])
        else:
            wkey.append(None)
        wkey = tuple(wkey)
        for word in phrasewords[:]:
            if word in self.stopwords \
            and not self.dbconn["cache"].has_key((wkey, word)):
                phrasewords.remove(word)
        if not phrasewords:
            for stopword in self.stopwords:
                if stopword in self.params["match_f"]:
                    phrasewords = [stopword]
                    break
        if not phrasewords:
            raise ERROR, "match_f ohne Wert!"
        froms.append('m')
        froms.append('t')
        if self.params.has_key("parse"):
            froms.append('s')
            ew =  "and m.satzid = s.satzid "
```

```
396                 pval = self.params["parse"]
397                 if pval == "NULL":
398                     ew = ew + "and s.parsequalitaet IS NULL "
399                 elif pval[:2] == "GT":
400                     ew = ew + "and s.parsequalitaet > %s " %pval[2:]
401                 else:
402                     ew = ew + "and s.parsequalitaet = %s " %pval
403             else:
404                 ew = ""
405             for word in phrasewords:
406                 debug("word%s\n" %‘(column, word, self.korpora)‘, 16)
407                 if self.dbconn["cache"].has_key((wkey, word)):
408                     currentids = self.dbconn["cache"][(wkey, word)]
409                     debug("currentids aus cache\n", 16)
410                 else:
411                     currentids = self.querydb(
412                         ("select distinct %s " %column)+\
413                         self.p_froms(froms) +\
414                         ("where t.name = ’%s’ " %word) +\
415                         ("and t.tokenid = m.tokenid ") +\
416                         ew +\
417                         ("%s" %self.korpora)
418                     )
419                     debug("currentids aus db\n", 16)
420                     self.dbconn["cache"][(wkey, word)] = currentids
421                 # currentids reduzieren auf Regionen, die nicht zu viele
422                 # Tokens enthalten
423                 if string.strip(column) in ["m.segmentnr"]:
424                     offset = 0
425                     csl = currentids[offset:offset+100]
426                     ncsl = []
427                     while csl:
428                         ncsl = ncsl + self.p_distinct(self.querydb(
429                             ("select %s " %column) +\
430                             ("from map_s_w m ") +\
431                             ("where m.segmentnr in (%s) " %(‘csl‘[1:-1])) +\
432                             ("group by m.satzid ") +\
433                             ("having count(m.tokenid) = %s" %wlen)
434                         ))
435                         offset = offset + 100
436                         csl = currentids[offset:offset+100]
437                     currentids = ncsl
438                     debug("currentids reduziert\n", 16)
439                     debug("currentids = %s\n" %‘currentids‘, 18)
440                 else:
441                     debug("currentids nicht reduziert\n", 16)
442                 # currentids mit vorhandener Liste schneiden
443                 if list:
444                     list = self.p_intersect(currentids, list)
445                     debug("geschnitten: %s\n" %‘list‘, 18)
446                 else:
447                     list = currentids         # erster Durchlauf
448                 if not list:                  # leerer Durchschnitt?
449                     debug("Schnitt bereits leer\n", 16)
450                     break                     # -> Ergebnis leer
451         elif self.params.has_key("match_l"):
452             phrasewords = self.params["match_l"][:]
453             for word in phrasewords[:]:
454                 if word in self.stoplemmata:
455                     phrasewords.remove(word)
456             if not phrasewords:
```

```
            for stopword in self.stoplemmata:
                if stopword in self.params["match_l"]:
                    phrasewords = [stopword]
                    break
        if not phrasewords:
            raise ERROR, "match_l ohne Wert!"
        phraselen = len(phrasewords)
        phrase= string.join(phrasewords)
        phrasewds = "('"+string.join(phrasewords,"','")+"')"
        froms.append('g')
        froms.append('m')
        froms.append('t')
        froms.append('s')
        if self.params.has_key("parse"):
            pval = self.params["parse"]
            if pval == "NULL":
                ew = "and s.parsequalitaet IS NULL "
            elif pval[:2] == "GT":
                ew = "and s.parsequalitaet > %s " %pval[2:]
            else:
                ew = "and s.parsequalitaet = %s " %pval
        else:
            ew = ""
        list = self.querydb(
            ("select distinct %s " %column) +\
            self.p_froms(froms) +\
            ("where g.name IN %s " %phrasewds) +\
            ("and t.tokenid = m.tokenid ") +\
            ("%s " %self.korpora)  +\
            ("and s.satzid = m.satzid ") +\
            ("and t.grundformid = g.grundformid ") +\
            ew +\
            ("group by m.satzid ") +\
            ("having count(m.tokenid) >= %s" %phraselen)
        )
    elif self.params.has_key("match_lf"):
        lemmatas = self.params["match_lf"][:]   # list
        lemlen = len(lemmatas)
        lkey = [column, self.korpora]
        if self.params.has_key("parse"):
            lkey.append(self.params["parse"])
        else:
            lkey.append(None)
        lkey = tuple(lkey)
        for word in lemmatas[:]:
            if word in self.stoplemmata \
            and not self.dbconn["cache"].has_key((lkey, word)):
                lemmatas.remove(word)
        if not lemmatas:
            for stopword in self.stoplemmata:
                if stopword in self.params["match_lf"]:
                    lemmatas = [stopword]
                    break
        if not lemmatas:
            raise ERROR, "match_lf ohne Wert!"
        froms.append('m')
        froms.append('t')
        froms.append('g')
        if self.params.has_key("parse"):
            froms.append('s')
            ew =  "and m.satzid = s.satzid "
```

```
518                 pval = self.params["parse"]
519                 if pval == "NULL":
520                     ew = ew + "and s.parsequalitaet IS NULL "
521                 elif pval[:2] == "GT":
522                     ew = ew + "and s.parsequalitaet > %s " %pval[2:]
523                 else:
524                     ew = ew + "and s.parsequalitaet = %s " %pval
525             else:
526                 ew = ""
527             for lemma in lemmatas:
528                 debug("lemma%s\n" %‘(column, lemma, self.korpora)‘, 16)
529                 if self.dbconn["cache"].has_key((lkey, lemma)):
530                     currentids = self.dbconn["cache"][(lkey, lemma)]
531                     debug("currentids aus cache\n", 16)
532                 else:
533                     currentids = self.querydb(
534                         ("select distinct %s " %column)+\
535                         self.p_froms(froms) +\
536                         ("where g.name = ’%s’ " %lemma) +\
537                         ("and t.tokenid = m.tokenid ") +\
538                         ("and g.grundformid = t.grundformid ") +\
539                         ew +\
540                         ("%s" %self.korpora)
541                     )
542                     debug("currentids aus db\n", 16)
543                     self.dbconn["cache"][(lkey, lemma)] = currentids
544                 # currentids reduzieren auf Regionen, die nicht zu viele
545                 # Tokens enthalten
546                 if string.strip(column) in ["m.segmentnr"]:
547                     offset = 0
548                     csl = currentids[offset:offset+100]
549                     ncsl = []
550                     while csl:
551                         ncsl = ncsl + self.p_distinct(self.querydb(
552                             ("select %s " %column) +\
553                             ("from map_s_w m ") +\
554                             ("where m.segmentnr in (%s) " %(‘csl‘[1:-1])) +\
555                             ("group by m.satzid ") +\
556                             ("having count(m.tokenid) = %s" %lemlen)
557                         ))
558                         offset = offset + 100
559                         csl = currentids[offset:offset+100]
560                     currentids = ncsl
561                     debug("currentids reduziert\n", 16)
562                     debug("currentids = %s\n" %‘currentids‘, 18)
563                 else:
564                     debug("currentids nicht reduziert\n", 16)
565                 # currentids mit vorhandener Liste schneiden
566                 if list:
567                     list = self.p_intersect(currentids, list)
568                 else:
569                     list = currentids         # erster Durchlauf
570                 if not list:                  # leerer Durchschnitt?
571                     debug("Schnitt bereits leer\n", 16)
572                     break                     # -> Ergebnis leer
573         elif self.params.has_key("lemmata")\
574         or   self.params.has_key("lem_inv"):
575             lkey = [column, self.korpora]
576             if self.params.has_key("parse"):
577                 lkey.append(self.params["parse"])
578             else:
```

```
579             lkey.append(None)
580         lkey = tuple(lkey)
581         if self.params.has_key("lemmata"):
582             lemmatas = self.params["lemmata"][:]   # list
583             for word in lemmatas[:]:
584                 if word in self.stoplemmata \
585                 and not self.dbconn["cache"].has_key((lkey, word)):
586                     lemmatas.remove(word)
587             if not lemmatas:
588                 for stopword in self.stoplemmata:
589                     if stopword in self.params["lemmata"]:
590                         lemmatas = [stopword]
591                         break
592             if not lemmatas:
593                 raise ERROR, "lemmata ohne Wert!"
594         else:
595             lemmatas = []
596         if self.params.has_key("lem_inv"):
597             lem_invs = self.params["lem_inv"]        # list
598         else:
599             lem_invs = []
600         froms.append('m')
601         froms.append('t')
602         froms.append('g')
603         if self.params.has_key("parse"):
604             froms.append('s')
605             ew =  "and m.satzid = s.satzid "
606             pval = self.params["parse"]
607             if pval == "NULL":
608                 ew = ew + "and s.parsequalitaet IS NULL "
609             elif pval[:2] == "GT":
610                 ew = ew + "and s.parsequalitaet > %s " %pval[2:]
611             else:
612                 ew = ew + "and s.parsequalitaet = %s " %pval
613         else:
614             ew = ""
615         list_leer = 0
616         for lemma in lemmatas:
617             debug("lemma%s\n" %'(column, lemma, self.korpora)', 16)
618             if self.dbconn["cache"].has_key((lkey, lemma)):
619                 currentids = self.dbconn["cache"][(lkey, lemma)]
620                 debug("currentids aus cache\n", 16)
621             else:
622                 currentids = self.querydb(
623                     ("select distinct %s " %column)+\
624                     self.p_froms(froms) +\
625                     ("where g.name = '%s' " %lemma) +\
626                     ("and t.tokenid = m.tokenid ") +\
627                     ("and g.grundformid = t.grundformid ") +\
628                     ew +\
629                     ("%s" %self.korpora)
630                 )
631                 debug("currentids aus db\n", 16)
632                 self.dbconn["cache"][(lkey, lemma)] = currentids
633             if list:
634                 list = self.p_intersect(currentids, list)
635             elif not list_leer:
636                 list = currentids         # erster Durchlauf
637             if not list:                  # leerer Durchschnitt?
638                 list_leer = 1
639                 debug("Schnitt bereits leer\n", 16)
```

```
                break                   # -> Ergebnis leer
            for lem_inv in lem_invs:
                debug("lem_inv%s\n" %‘(column, lem_inv, self.korpora)‘, 16)
                if self.dbconn["cache"].has_key((lkey, lem_inv)):
                    currentids = self.dbconn["cache"][(lkey, lem_inv)]
                    debug("currentids aus cache\n", 16)
                else:
                    currentids = self.querydb(
                        ("select distinct %s " %column)+\
                        self.p_froms(froms) +\
                        ("where g.reverse LIKE ’%s%%’ " %lem_inv) +\
                        ("and t.tokenid = m.tokenid ") +\
                        ("and g.grundformid = t.grundformid ") +\
                        ew +\
                        ("%s" %self.korpora)
                    )
                    debug("currentids aus db\n", 16)
                    self.dbconn["cache"][(lkey, lem_inv)] = currentids
                if list:
                    list = self.p_intersect(currentids, list)
                elif not list_leer:
                    list = currentids         # erster Durchlauf
                if not list:                  # leerer Durchschnitt?
                    list_leer = 1
                    debug("Schnitt bereits leer\n", 16)
                    break                     # -> Ergebnis leer
        elif self.params.has_key("parse"):    # parse alleine
            froms.append(’s’)
            froms.append(’m’)
            pval = self.params["parse"]
            if pval == "NULL":
                ew = "and s.parsequalitaet IS NULL "
            elif pval[:2] == "GT":
                ew = "and s.parsequalitaet > %s " %pval[2:]
            else:
                ew = "and s.parsequalitaet = %s " %pval
            if self.params.has_key("max"):
                limit = "LIMIT %s " %self.params["max"]
            else:
                limit = ""
            list = self.querydb(
                ("select distinct %s " %column) +\
                self.p_froms(froms) +\
                ("where m.satzid = s.satzid ") +\
                ew +\
                ("%s" %self.korpora) +\
                limit
            )
        elif self.params.has_key("beispiel zum segment mit id") \
        or   self.params.has_key("bid"):
            try:
                if self.params.has_key("bid"):
                    bid = self.params["bid"]
                else:
                    bid = self.params["beispiel zum segment mit id"]
                bid = int(bid)
            except ValueError:
                raise ERROR, "bid keine ganze Zahl"
            if self.params.has_key("max"):
                limit = "LIMIT %s " %self.params["max"]
            else:
```

```
            limit = ""
        froms.append('m')
        froms.append('ph_e')
        froms.append('ph_c')
        if self.params.has_key("parse"):
            froms.append('s')
            ew =  "and m.satzid = s.satzid "
            pval = self.params["parse"]
            if pval == "NULL":
                ew = ew + "and s.parsequalitaet IS NULL "
            elif pval[:2] == "GT":
                ew = ew + "and s.parsequalitaet > %s " %pval[2:]
            else:
                ew = ew + "and s.parsequalitaet = %s " %pval
        else:
            ew = ""
        list = self.querydb(
            ("select distinct %s " %column) +\
            self.p_froms(froms) +\
            ("where m.segmentnr = ph_e.segmentnr ") +\
            ("and ph_e.pc_id = ph_c.pc_id ") +\
            ("and ph_c.segmentnr = %s " %bid) +\
            ew +\
            ("%s" %self.korpora) +\
            limit
        )
    elif self.params.has_key("phrases"):
        try:
            sent = self.params["phrases"]
            lang = self.params["lang"]
        except KeyError:
            raise ERROR, "Sprache wurde nicht angegeben"
        wort = ""
        for m in ["wort", "interessant", "i"]:
            wstart = string.find(sent, "<%s>"  %m)
            if wstart < 0:
                continue
            wende  = string.find(sent, "</%s>" %m, wstart)
            if wstart >= 0 and wende >= 0:
                if wort:
                    raise ERROR, "mehrere Markierungen nicht unterstuetzt"
                wstart = wstart + 2 + len(m)
                wort = string.strip(sent[wstart:wende])
                debug("\twort:"+`wort`+"\n", 10)
            elif wstart >= 0:
                raise ERROR, "</%s> nach <%s> nicht gefunden" %(m,m)
        tagged = self.p_tagline(sent, lang)
        markiert = range(len(tagged.getWoerter()))      # 2do
        debug("\ttagged:"+`tagged.getWoerter()`+"\n",10)
        for qdb_type in ["erkannt", "wb"]:
            debug("\tqdb_type:"+`qdb_type`+"\n",10)
            #global debug_level
            #dl = debug_level
            try:
                #debug_level = -5
                filter = self.filter[(lang, qdb_type)]
                #debug_level = dl
            except KeyError:
                raise ERROR, "Sprache %s unbekannt" %`lang`
            #debug_level = -5
            phrases = filter[(markiert, tagged)]
```

```
762                 #debug_level = dl
763                 debug("\tphrases.keys():"+‘phrases.keys()‘+"\n", 10)
764                 atoms = tagged.getRegionenMit(atom.Atom)
765                 for part in phrases.keys():
766                     src_dst_liste = phrases[part]
767                     if len(src_dst_liste) > 0:
768                         ausgangspkt = atoms[part]
769                         try:
770                             if ausgangspkt.getDelimiter() == atom.Atom.delimiter:
771                                 ausgangspkt = region.Regionen([ausgangspkt])
772                         except AttributeError:
773                             pass
774                         debug("\taphrase:"+‘ausgangspkt.getText()‘+"\n", 10)
775                         if (not wort) \
776                         or (string.find(ausgangspkt.getText(), wort) >= 0):
777                              for (src, dst) in src_dst_liste:
778                                  src.qdb_type = qdb_type
779                                  list.append(src)
780                                  debug("\tlist.append(%s)\n" %‘src‘, 20)
781                         #    ausgangspkt.qdb_type = qdb_type
782                         #    list.append(ausgangspkt)
783         elif self.params.has_key("tags"):
784             try:
785                 lang = self.params["lang"]
786                 max = int(self.params["max"])
787             except KeyError:
788                 raise ERROR, "Sprache oder Max wurde nicht angegeben"
789             taglang = "tag_" + lang
790             lkey = (column, self.korpora)
791             tags = self.params["tags"][:]
792             froms.append(’m’)
793             froms.append(’t’)
794             froms.append(taglang)
795             while len(list) < max:
796                 firsttag = tags[0]
797                 offset = 0
798                 idlist = self.querydb(
799                         ("select %s " %column)+\
800                         self.p_froms(froms) +\
801                         ("where %s.name = ’%s’ " %(taglang, firsttag)) +\
802                         ("and %s.tagid = t.tagid " %taglang) +\
803                         ("and t.tokenid = m.tokenid ") +\
804                         ("%s" %self.korpora) +\
805                         ("LIMIT %s, %s" %(offset, 2 * max))
806                     )
807                 offset = offset + (2 * max)
808                 idlist = self.p_uniquify(idlist)
809                 debug("currentids aus db%s\n" %idlist, 16)
810                 for moretag in tags[1:]:
811                     idlist = self.querydb(
812                         ("select %s " %column)+\
813                         self.p_froms(froms) +\
814                         ("where %s.name = ’%s’ " %(taglang, moretag)) +\
815                         ("and %s.tagid = t.tagid " %taglang) +\
816                         ("and t.tokenid = m.tokenid ") +\
817                         ("and %s in (%s)" %(column, ‘idlist‘[1:-1])) +\
818                         ("%s" %self.korpora)
819                     )
820                     idlist = self.p_uniquify(idlist)
821                     debug("currentids aus db%s\n" %idlist, 16)
822                 list = list + idlist
```

```
823         debug("list: " + ‘list‘ + "\n", 18)
824         return list
825 
826     def p_uniquify(self, list):
827         nlist = []
828         for item in list:
829             if item in nlist:
830                 continue
831             nlist.append(item)
832         return nlist
833 
834     def p_tagline(self, line, lang):
835         return tagger.tagline(line, lang)
836 
837     def p_dropwords(self, list):
838         matchkey = ""
839         for key in ["match", "match_l", "match_f", "match_lf", "tags"]:
840            if self.params.has_key(key):
841                 matchkey = key
842                 break
843         strict = self.params.has_key("strict")
844         if self.params.has_key("lemmata"):
845             list = self.p_dropwords_lemmata(list, strict)
846         elif self.params.has_key("lem_inv"):
847             list = self.p_dropwords_lem_inv(list, strict)
848         elif self.params.has_key("tags"):
849             list = self.p_dropwords_tag(list, strict)
850         elif matchkey:
851             list = self.p_dropwords_match(list, matchkey, strict)
852         return list
853 
854     def p_dropwords_lemmata(self, list, strict):
855         return self.p_dropwords_match(list, "lemmata", strict)
856 
857     def p_dropwords_lem_inv(self, list, strict):
858         return self.p_dropwords_match(list, "lem_inv", strict)
859 
860     def p_dropwords_tag(self, list, strict):
861         return self.p_dropwords_match(list, "tags", strict)
862 
863     def p_dropwords_match(self, list, matchkey, strict):
864         phrasewords = self.params[matchkey]
865         debug("phrasewords=%s\n" %‘phrasewords‘, 11)
866         debug("matchkey   =%s\n" %‘matchkey‘,    12)
867         debug("strict     =%s\n" %‘strict‘,      12)
868         #if len(phrasewords) <= 1:
869         #    return list             # nichts zu droppen
870         # sehr wohl! bei match* sind die Segmente zu verwerfen, die
871         # zusaetzliche Woerter enthalten
872         neulist = []
873         segmentliste = []
874         for index in range(len(list)):
875             neu = index % len(self.langids)
876             segmentliste.append(list[index])
877             if neu == (len(self.langids)-1):
878                 uebernehmen = 0
879                 for segment in segmentliste:
880                     if matchkey in ["match", "match_f"]:
881                         words = segment.getWoerter()
882                     elif matchkey in ["match_l", "match_lf", "lemmata", "lem_inv"]:
883                         words = segment.getLemmata()
```

```
                elif matchkey in ["tags"]:
                    words = segment.getTags()
                else:
                    raise ERROR, "%s nicht vollstaendig implementiert" %matchkey
                debug("words      =%s\n" %‘words‘, 20)
                verworfen = 0
                if strict:
                    #todo: fuer lem_inv implementieren!
                    if matchkey in [
                        "match", "match_f", "match_l", "match_lf",
                        "lemmata", "tags"
                    ]:
                        strictphrase = string.join(phrasewords, " ")
                        wholephrase = string.join(words, " ")
                        debug("strict: %s, whole: %s" %(strictphrase, wholephrase), 99)
                        if string.find(wholephrase, strictphrase) == -1:
                            verworfen = 1
                            debug("verworfen wg. strict!\n", 12)
                            break
                    else:
                        raise ERROR, "strict fuer %s nicht vollstaendig implementiert" %matchkey
                # wenn durch strict durchgegangen, koennen immernoch
                # zuviele Woerter drin sein, also neues if
                if matchkey in ["match", "match_f", "match_l", "match_lf"]:
                    for overfl in words:
                        if not overfl in phrasewords:
                            verworfen = 1
                            debug("verworfen wg. overfl=%s\n" %overfl, 12)
                            break
                elif matchkey in ["lemmata", "tags"]:
                    for notwendig in phrasewords:
                        if not notwendig in words:
                            verworfen = 1
                            debug("verworfen wg. notw.=%s\n" %notwendig, 12)
                            break
                elif matchkey in ["lem_inv"]:
                    for notwendig in phrasewords:
                        n_inv = string.lower(reverse(notwendig))
                        n_in_words = 0
                        for lem in words:
                            if string.find(string.lower(lem),
                            string.lower(n_inv)) + len(n_inv) == len(lem):
                                n_in_words = 1
                                break
                        if not n_in_words:
                            verworfen = 1
                            debug("verworfen wg. notw.=%s\n" %notwendig, 12)
                            break
                else:
                    raise NotImplementedError
                laenge_ausreichend = len(words) >= len(phrasewords)
                if not verworfen and laenge_ausreichend:
                    debug("Uebernahme\n", 12)
                    uebernehmen = 1
            if uebernehmen:
                for segment in segmentliste:
                    neulist.append(segment)
            segmentliste = []
        for i in neulist:
            debug("neulist[%s]=%s\n" %(‘i‘,i.getText()), 25)
        return neulist
```

```
945
946
947 class QuerySatz(QueryPhrase):
948
949     def __init__(self, dbconn, line, usekorpora):
950         QueryPhrase.__init__(self, dbconn, line, usekorpora, ["id", "wort",
951         "lemma", "match", "match_f", "lemmata", "lem_inv", "parse", "setparse",
952         "match_l", "match_lf", "_fs", "tags", "strict",
953         "beispiel zum segment mit id", "bid"])
954         self.query_type = "satz"
955
956     def real_query(self):
957         list = self.p_query(" m.satzid ")
958         if self.params.has_key("max"):
959             list = list[:int(self.params["max"])]
960         if self.params.has_key("setparse"):          # Parsequalitaet setzen
961             value = self.params["setparse"]
962             for id in list:
963                 self.dbconn["query"].update(
964                     ("update saetze ") +\
965                     ("set parsequalitaet = %s " %value) +\
966                     ("where satzid = %s" %‘id‘))
967             self.dbconn["cache"].invalidate()
968             return "Parsequalitaet gesetzt"
969         satzlist = []
970         for id in list:
971             satzlist.append(satzdb.SatzVonDb(id, self.dbconn["regionen"]))
972         return self.p_dropwords(satzlist)
973
974
975 class QuerySegment(QueryPhrase):
976
977     def __init__(self, dbconn, line, usekorpora):
978         QueryPhrase.__init__(self, dbconn, line, usekorpora, ["id", "wort",
979         "lemma", "match", "match_f", "lemmata", "lem_inv", "lang", "satzid",
980         "phrases", "parse", "match_l", "match_lf", "_fs", "tags", "strict",
981         "beispiel zum segment mit id", "bid"])
982         self.query_type = "segment"
983         self.langids = config.lang2id.values()
984         langmap = {}
985         langmap["deutsch"]  = config.lang2id["de"]
986         langmap["englisch"] = config.lang2id["en"]
987         if self.params.has_key("lang"):
988             langstr = self.params["lang"]
989             if langmap.has_key(langstr):
990                 self.langids = [langmap[langstr]]
991             elif config.lang2id.has_key(langstr):
992                 self.langids = [config.lang2id[langstr]]
993             elif langstr in config.lang2id.values():
994                 self.langids = [langstr]
995             else:
996                 raise ERROR, "querydb.QuerySegment: Unbekannte Sprache "+‘langstr‘
997
998     def real_query(self):
999         segmentlist = []
1000         list = self.p_query(" m.segmentnr ")
1001         if self.params.has_key("max"):
1002             list = list[:int(self.params["max"])]
1003         if self.params.has_key("phrases"):
1004             return list
1005         for id in list:
```

```
            for lang in self.langids:
                segmentlist.append(segmentdb.SegmentVonDb(id, self.dbconn["regionen"], lang))
        return self.p_dropwords(segmentlist)

class Query(zeitmessung.Zeitmessung):

    def __init__(self, regionenDbconn=None, dbg=-1):
        zeitmessung.Zeitmessung.__init__(self)
        if dbg >= 0:
            global debug_level
            debug_level = dbg
        self.dbconn = {}
        self.dbconnCloseOnExit = []
        self.dbconn["query"] = db.Db()
        self.dbconnCloseOnExit.append("query")
        if regionenDbconn:
            self.dbconn["regionen"] = regionenDbconn
        else:
            self.dbconn["regionen"] = db.Db()
            self.dbconnCloseOnExit.append("regionen")
        self.closed = 0
        self.setCache(cache.DefaultCache(self))
        self.setServer(None)
        self.setClientID(-1)
        self.selects = [ "text", "pyob", "all", "id", "count",
                         "debug",
                         "_r", "_r80", "_lr", "_lr80",
                         "_s", "_s80", "_ls", "_ls80"
                       ]
        self.types = ["wort", "grundform", "satz", "segment"]
        self.usekorpora = ""
        self.ETB   = "<DONE>\n"
        self.ERROR = "<ERROR>\n"
        self.unpack={}
        self.unpack[None]    = UnpackError(self.ERROR, self.ETB)   # immer ERROR
        self.unpack["all"]   = UnpackListRepr(self.ERROR, self.ETB)
        self.unpack["count"] = UnpackCount(self.ERROR, self.ETB)
        self.unpack["debug"] = UnpackListDebug(self.ERROR, self.ETB)
        self.unpack["id"]    = UnpackListID(self.ERROR, self.ETB)
        self.unpack["pyob"]  = UnpackPyob(self.ERROR, self.ETB)
        self.unpack["text"]  = UnpackText(self.ERROR, self.ETB)
        self.unpack["_lr"]   = UnpackListRepr(self.ERROR, self.ETB)
        self.unpack["_lr80"] = UnpackListRepr80(self.ERROR, self.ETB)
        self.unpack["_ls"]   = UnpackListString(self.ERROR, self.ETB)
        self.unpack["_ls80"] = UnpackListString80(self.ERROR, self.ETB)
        self.unpack["_r"]    = UnpackRepr(self.ERROR, self.ETB)
        self.unpack["_r80"]  = UnpackRepr80(self.ERROR, self.ETB)
        self.unpack["_s"]    = UnpackString(self.ERROR, self.ETB)
        self.unpack["_s80"]  = UnpackString80(self.ERROR, self.ETB)

    def setCache(self, cache):
        self.cache = cache
        self.dbconn["cache"] = cache

    def setServer(self, server):
        self.server = server

    def setClientID(self, cid):
        self.cid = cid

    def __getstate__(self):
```

```
        return (self.usekorpora,)

    def close(self):
        if self.closed:
            sys.stderr.write("Error: trying to close query already closed\n")
        for dbconnKey in self.dbconnCloseOnExit:
            self.dbconn[dbconnKey].close()
            del self.dbconn[dbconnKey]    # damit spaeter Verwendung auffaellt
        self.dbconnCloseOnExit = []
        self.closed = 1

    def p_makeQuery(self, type, where, restricted = 0):
        q_obj = None
        if type == "wort" or type == "grundform":
            q_obj = QueryWort(self.dbconn, where, self.usekorpora, type)
        elif type == "satz" and not restricted:
            q_obj = QuerySatz(self.dbconn, where, self.usekorpora)
        elif type == "segment" and not restricted:
            q_obj = QuerySegment(self.dbconn, where, self.usekorpora)
        return q_obj

    def p_checkStart(self, starts, vars):
        for i in range(0, len(vars)):
            var = vars[i]
            stw = starts[i]
            if var[0:2] != stw:
                raise ERROR, "bad command!"
            vars[i] = var[2:]
        return vars

    def query(self, line, restricted=0):
        usage = "bad command!"
        q_line = string.split(line, ";")
        if len(q_line) == 1 and q_line[0][0:2] == "u:":
            corpora = string.lower(q_line[0][2:])
            if corpora == "wb":
                corpora = config.wbids
            elif corpora == "erkannt":
                corpora = "(-1)"
            elif corpora == "wb+erkannt":
                corpora = config.wbids[:-1] + ", -1)"
            elif corpora == "good":
                corpora = config.goodcorpora
            elif corpora in ["none", "all"]:
                corpora = None
            elif corpora[:1] == "(":
                if corpora[-1:] != ")":
                    return ("schliessende Klammer fehlt", None)
            else:
                return ("Korpusart %s unbekannt" %corpora, None)
            if corpora:
                self.usekorpora = "and s.satzid = m.satzid " +\
                                  "and s.herkunftsid in %s" %(corpora)
                return ("using corpora %s" %corpora, "_s")
            else:
                self.usekorpora = ""
                return ("using all corpora", "_s")
        else:
            if len(q_line) != 3:
                return (usage, None)
            else:
```

```
                try:
                    type, select, where = self.p_checkStart(["t:", "s:", "w:"], q_line)
                    if not select in self.selects or not type in self.types:
                        return (usage, None)
                    q_obj = self.p_makeQuery(type, where)
                    qkey = ‘q_obj‘
                    debug("\tqkey = %s\n" %qkey, 11)
                    try:
                        temp = self.cache[qkey]
                        list, max = temp
                        # 2do: immer nur bisher laengste Liste cachen
                        #      -> ’max’ aus q_obj.__repr__() entfernen
                        #for i in list:
                        #    if isinstance(i, regiondb.RegionVonDb) \
                        #    or isinstance(i, atomdb.AtomVonDb):
                        #        i.setDbconn(self.dbconn["query"])       # 2do
                    except KeyError:
                        list, max = q_obj.query(restricted, select)
                        self.cache[qkey] = (list, max)
                    except TypeError:
                        debug("Error: %s in cache\n" %‘temp‘)
                        list, max = q_obj.query(restricted, select)
                        self.cache[qkey] = (list, max)
                    q_obj.close()
                    return (list[:max], select)
                except ERROR, data:
                    return (data, None)
                except MySQLdb.OperationalError, data:
                    return ("Datenbankfehler %s" %data, None)

    def serve(self, port):
        debug(’Query.serve() called, instead use QueryRequestHandler!\n’)
        sock = socket(AF_INET, SOCK_STREAM)
        sock.bind(("", port))
        sock.listen(1)
        debug(’Service started on %s\n’ % time.ctime(time.time()))
        stop = 0
        while not stop:
            conn, addr = sock.accept()
            stop = thread.start_new_thread(self.run, (conn, addr))
        sock.close()      # Was mach sock.shutdown() ?
        debug(’Service stopped on %s\n’ %time.ctime(time.time()))

    def info(self, beschreibung):
        self.server.infoMutex.acquire()
        self.server.info[self.cid] = beschreibung
        self.server.infoMutex.release()

    def infodone(self):
        self.server.infoMutex.acquire()
        self.server.info[self.cid] = self.server.info[self.cid] + " - done"
        self.server.infoMutex.release()


    def serverJob(self, name, *param):    # Jobs, die der Server erledigen muss
        self.p_begin("serverJob")
        self.info("beauftrage Server mit %s" %name)
        job = [self.cid, name]
        for i in param:
            job.append(i)
        self.server.jobsMutex.acquire()
```

```
1189            self.server.jobs.append(job)
1190            self.server.jobsMutex.release()
1191            fertig = 0
1192            while not fertig:
1193                time.sleep(1.0)
1194                self.server.jobsMutex.acquire()
1195                fertig = 1
1196                for i in self.server.jobs:
1197                    if len(i) and i[0] == job[0]:
1198                        fertig = 0
1199                        break
1200                self.server.jobsMutex.release()
1201            self.infodone()
1202            self.p_end("serverJob")
1203            return "%s done" %name      # 2do: Antwort vom Server holen
1204
1205        def cmd_debug(self, data, conn, restricted):
1206            self.p_begin("cmd_debug")
1207            self.info("command " + ‘data‘)
1208            global debug_level
1209            if string.strip(data[5:]):
1210                debug_level = int(data[5:])
1211                query = "debug level set to %s" %‘debug_level‘
1212            else:
1213                query = "debug level is %s" %‘debug_level‘
1214            self.unpack["_s"](conn.send, query, restricted)
1215            self.unpack["_s"](debug, query, restricted)
1216            self.infodone()
1217            self.p_end("cmd_debug")
1218            return 1
1219
1220        def cmd_cache(self, data, conn, restricted):
1221            self.p_begin("cmd_cache")
1222            self.info("command " + ‘data‘)
1223            if data == "cache.clear":
1224                self.cache.invalidate()
1225                query = "cache cleared"
1226                self.unpack["_s"](conn.send, query, restricted)
1227                self.unpack["_s"](debug, query, restricted)
1228            elif data[:19] == "cache.autoclear mem":
1229                mem = string.strip(data[19:])
1230                try:
1231                    self.server.ac_mem = int(mem)
1232                except:
1233                    query = "number expected, found %s" %mem
1234                self.server.ac_mem2 = 0
1235                query = "mem set to %s KB" %mem
1236                self.unpack["_s"](conn.send, query, restricted)
1237                self.unpack["_s"](debug, query, restricted)
1238            elif data[:20] == "cache.autoclear time":
1239                zeit = string.strip(data[20:])
1240                try:
1241                    sekunden = int(zeit)
1242                except:
1243                    query = "number expected, found %s" %zeit
1244                if sekunden < 60:
1245                    query = "number of seconds must be greater than or equal to 60"
1246                else:
1247                    self.server.ac_trdist = sekunden
1248                    self.server.ac_tcdist = sekunden / 2
1249                    query = "time set to %s seconds" %sekunden
```

```
1250            self.unpack["_s"](conn.send, query, restricted)
1251            self.unpack["_s"](debug, query, restricted)
1252        elif data[:10] == "cache.save":
1253            fname = "qdb_"+string.strip(data[10:])+".cache"
1254            try:
1255                f=open(fname, "w")
1256                pickle.dump(self.cache, f)
1257                f.close()
1258                query = "cache saved as %s" %fname
1259            except IOError:
1260                query = "could not write to %s" %fname
1261            self.unpack["_s"](conn.send, query, restricted)
1262            self.unpack["_s"](debug, query, restricted)
1263        elif data[:10] == "cache.load":
1264            fname = "qdb_"+string.strip(data[10:])+".cache"
1265            ncache = None
1266            query = "cache not changed"
1267            try:
1268                f=open(fname, "r")
1269                ncache = pickle.load(f)
1270                f.close()
1271            except IOError:
1272                query = "unable to read from %s" %fname
1273            if ncache:
1274                if not self.server:
1275                    self.setCache(ncache)
1276                    query = "%s.cache loaded from %s" %(self,fname)
1277                else:
1278                    query = self.serverJob("setCache", ncache)
1279            self.unpack["_s"](conn.send, query, restricted)
1280            self.unpack["_s"](debug, query, restricted)
1281        elif data[:9] == "cache.add":
1282            fname = "qdb_"+string.strip(data[9:])+".cache"
1283            ncache = None
1284            query = "cache not changed"
1285            try:
1286                f=open(fname, "r")
1287                ncache = pickle.load(f)
1288                f.close()
1289            except IOError:
1290                query = "unable to read from %s" %fname
1291            if ncache:
1292                num = 0
1293                for key in ncache.keys():
1294                    self.cache[key] = ncache[key]
1295                    num = num + 1
1296                query = "%s items added to %s.cache from %s" %(num,self,fname)
1297            self.unpack["_s"](conn.send, query, restricted)
1298            self.unpack["_s"](debug, query, restricted)
1299        elif data[:12] == "cache.delete":
1300            query = "no line deleted"
1301            try:
1302                line = string.strip(data[12:])
1303                q_line = string.split(line, ";")
1304                type, select, where = self.p_checkStart(["t:", "s:", "w:"], q_line)
1305                q_obj = self.p_makeQuery(type, where)
1306                line = ‘q_obj‘
1307                if self.cache.has_key(line):
1308                    del self.cache[line]
1309                    query = "line %s deleted from cache" %(line)
1310                column = None
```

```
            if type == "satz":
                column = " m.satzid "
            if type == "segment":
                column = " m.segmentnr "
            ckey = []
            if q_obj.params.has_key("match_f"):
                ckey.append(1)
            if q_obj.params.has_key("match_f") \
            or q_obj.params.has_key("match_lf"):
                ckey.append(column)
                ckey.append(q_obj.korpora)
                if q_obj.params.has_key("parse"):
                    ckey.append(q_obj.params["parse"])
                else:
                    ckey.append(None)
                ckey = tuple(ckey)
            if q_obj.params.has_key("lemmata") \
            or q_obj.params.has_key("lem_inv"):
                query = query + "\nWarnung: derived lines not deleted"
            words = []
            if q_obj.params.has_key("match_lf"):
                words = q_obj.params["match_lf"]
            if q_obj.params.has_key("match_f"):
                words = q_obj.params["match_f"]
            for word in words:
                if self.cache.has_key((ckey, word)):
                    del self.cache[(ckey, word)]
                    query = query + "\nderived line %s deleted" %‘(ckey, word)‘
        except ERROR:
            pass
        self.unpack["_s"](conn.send, query, restricted)
        self.unpack["_s"](debug, query, restricted)
    elif data == "cache.stat":
        query = self.cache.statistic()
        self.unpack["_s"](conn.send, query, restricted)
        self.unpack["_s"](debug, query, restricted)
    elif data == "cache.keys":
        query = self.cache.keys()
        self.unpack["_lr"](conn.send, query, restricted)
        self.unpack["_lr"](debug, query, restricted)
    elif data == "cache.items":
        query = []
        for (key, value) in self.cache.items():
            if type(key) == types.StringType:
                kstr = key
            else:
                kstr = ‘key‘
            if type(value) == types.ListType:
                vstr = "IDs "+‘value‘
            elif type(value) == types.TupleType:
                try:
                    l,m = value
                    l2 = []
                    for reg in l:
                        l2.append(reg.regionid)
                    vstr = "Regs " + ‘l2‘
                except:
                    vstr = ‘value‘
            else:
                vstr = ‘value‘
            query.append(kstr+’\t’+vstr)
```

```
            self.unpack["_ls80"](conn.send, query, restricted)
            self.unpack["_ls"](debug, query, restricted)
        else:
            self.info("bad command " + ‘data‘)
            self.infodone()
            self.p_end("cmd_cache")
            return 0
        self.infodone()
        self.p_end("cmd_cache")
        return 1

    def cmd_info(self, data, conn, restricted):
        self.p_begin("cmd_info")
        self.info("command " + ‘data‘)
        arg = string.strip(data[4:])
        if arg == "":
            self.server.infoMutex.acquire()
            query = self.server.info.keys()
            self.server.infoMutex.release()
            self.unpack["_r"](conn.send, query, restricted)
        else:
            arglst = string.split(arg)
            query = "Argument erwartet"
            if "*" in arglst:
                arglst = []
                self.server.infoMutex.acquire()
                for i in self.server.info.keys():
                   arglst.append(‘i‘)
                self.server.infoMutex.release()
                query = "kein Client-Thread vorhanden"
            if len(arglst) == 0:
                self.unpack[None](conn.send, query, restricted)
            elif len(arglst) == 1:
                self.server.infoMutex.acquire()
                try:
                    query = self.server.info[int(arglst[0])]
                except:
                    query = "no info for %s" %arglst[0]
                self.server.infoMutex.release()
                self.unpack["_s"](conn.send, query, restricted)
            else:
                query = []
                for arg in arglst:
                    self.server.infoMutex.acquire()
                    try:
                        query.append(arg + ": " + self.server.info[int(arg)])
                    except:
                        query.append(arg + ": " + "no info")
                    self.server.infoMutex.release()
                self.unpack["_ls"](conn.send, query, restricted)
        self.infodone()
        self.p_end("cmd_info")
        return 1

    def cmd_status(self, data, conn, restricted):
        self.p_begin("cmd_status")
        self.info("command " + ‘data‘)
        arg = string.strip(data[6:])
        slist = ["time", "dbname", "dbuser", "cwd", "user", "user_env",
                 "dbconnCloseOnExit",
                 "dbconn", "closed", "selects", "types", "usekorpora",
```

```
1433             "cache", "server", "cid",
1434             "server.start", "server.port", "version",
1435             "server.ac_tclast", "server.ac_trlast",
1436             "server.ac_tcdist", "server.ac_trdist",
1437             "server.ac_mem", "server.ac_mem2",
1438             "server.tlast", "server.tcount",
1439             "size", "pid",
1440            ]
1441        slist.sort()
1442        if arg == "":
1443            query = "OK"
1444            self.unpack["_s"](conn.send, query, restricted)
1445        elif not restricted:
1446            if arg == "*":
1447                args = slist
1448            elif arg in slist:
1449                args = [arg]
1450            elif arg == "db":
1451                args = ["dbname", "dbuser"]
1452            else:
1453                args = string.split(arg)
1454            query = []
1455            for arg in args:
1456                if arg == "time":
1457                    query.append("time server:")
1458                    self.server.zeitMutex.acquire()
1459                    s = self.server.statistic()
1460                    for key in s.keys():
1461                        query.append("%s: %s" %(key, `s[key]`))
1462                    self.server.zeitMutex.release()
1463                    query.append("time self:")
1464                    s = self.statistic()
1465                    for key in s.keys():
1466                        query.append("%s: %s" %(key, `s[key]`))
1467                elif arg == "version":
1468                    global version
1469                    query.append("version = %s" %version)
1470                elif arg == "cwd":
1471                    query.append("cwd = '%s'" %os.getcwd())
1472                elif arg == "dbname":
1473                    query.append("dbname = '%s'" %config.dbname)
1474                elif arg == "dbuser":
1475                    query.append("dbuser = '%s'" %config.dbuser)
1476                elif arg == "user_env":
1477                    query.append("user_env = '%s'" %os.environ["USER"])
1478                elif arg == "user":
1479                    query.append("user = '%s'" %config.user)
1480                elif arg == "cache":
1481                    ci = {}
1482                    ci["num_keys"] = len(self.cache.keys())
1483                    ci["id"] = id(self.cache)
1484                    ci["statistic"] = self.cache.statistic()
1485                    query.append("cache = '%s'" %`ci`)
1486                elif arg == "dbconn":
1487                    ci = {}
1488                    for key in self.dbconn.keys():
1489                        if key != "cache":
1490                            ci[key] = self.dbconn[key]
1491                    query.append("dbconn = '%s'" %`ci`)
1492                elif arg == "size":
1493                    size = getVmSize()
```

```
                    query.append("size = %s" %size)
                elif arg == "pid":
                    pid = os.getpid()
                    query.append("pid = %s" %pid)
                elif arg in slist:
                    obj = self
                    for attr in string.split(arg, '.'):
                        try:
                            obj = getattr(obj, attr)
                        except AttributeError:
                            query.append("%s.%s not found" %('obj',attr))
                    query.append("%s = %s" %(arg, 'obj'))
                else:
                    debug("cmd_status %s unbekannt\n" %'arg', 10)
                    self.infodone()
                    self.p_end("cmd_status")
                    return 0
            self.unpack["_ls"](conn.send, query, restricted)
        else:
            debug("cmd_status %s verweigert\n" %'arg', 10)
            self.infodone()
            self.p_end("cmd_status")
            return 0
        self.infodone()
        self.p_end("cmd_status")
        return 1

    def run(self, conn, addr):
        self.p_begin("run")
        debug('Connected by %s\n' %'addr', -1)
        key = secret.getKey()
        conn.send("Was ist secret.getValue(" + 'key' + ")?\n")
        data = string.strip(conn.recv(1024))
        if not secret.valueOK(key, data):
          conn.send("Das ist leider falsch. "
          "Sie erhalten nur Zugriff auf das Woerterbuch.\n")
          debug('\tAuthentication failed, restricted access only\n', 0)
          restricted = 1
        else:
          conn.send("OK\n")
          debug('\tAuthentication OK for %s\n' %'addr', 0)
          restricted = 0
        stop = 0
        while 1:
            self.p_begin("recv")
            self.info("waiting for command")
            data = conn.recv(1024)
            self.infodone()
            self.p_end("recv")
            self.p_begin("data")
            if not data :              # Client hat Verbindung geschlossen
                debug('Connection %s closed by client.\n' %'addr', -1)
                break
            data = string.strip(data)
            if not data:               # Zeilen nur mit Whitespace ignorieren
                continue
            debug(("\t%s->" %'addr') + data + "\n", 3)
            if data == "quit":
                conn.send("bye\n%s" %self.ETB)
                break
            elif data == "shutdown" and not restricted:
```

```
                conn.send("stopping service\n")
                conn.send("this may take a few seconds\n%s" %self.ETB)
                stop = 1
                break
            elif data[:5] == "cache" and not restricted:
                if not self.cmd_cache(data, conn, restricted):
                    self.p_begin("unpack")
                    self.unpack[None](conn.send, "bad command", restricted)
                    self.p_end("unpack")
            elif data[:5] == "debug" and not restricted:
                if not self.cmd_debug(data, conn, restricted):
                    self.p_begin("unpack")
                    self.unpack[None](conn.send, "bad command", restricted)
                    self.p_end("unpack")
            elif data[:4] == "info" and not restricted:
                if not self.cmd_info(data, conn, restricted):
                    self.p_begin("unpack")
                    self.unpack[None](conn.send, "bad command", restricted)
                    self.p_end("unpack")
            elif data[:6] == "status":
                if not self.cmd_status(data, conn, restricted):
                    self.p_begin("unpack")
                    self.unpack[None](conn.send, "bad command", restricted)
                    self.p_end("unpack")
            else:
                self.info("query " + data)
                self.p_begin("query")
                query, select = self.query(data, restricted)
                self.p_end("query")
                self.info(data + " - unpacking")
                self.p_begin("unpack")
                self.unpack[select](conn.send, query, restricted)
                self.p_end("unpack")
                self.infodone()
            self.p_end("data")
        self.p_end("data")
        conn.close()
        debug('Connection to %s closed\n' %'addr', -1)
        self.info("closed")
        self.p_end("run")
        self.server.zeitMutex.acquire()
        self.server.p_addToStatistic(self.statistic())
        self.server.zeitMutex.release()
        return stop


class Unpack:

    def __init__(self, error_str, etb_str):
        self.ERROR = error_str
        self.ETB   = etb_str
        self.SelectError = "SelectError"

    def __call__(self, fun, query, restricted = 0):
        try:
            self.p_call(fun, query, restricted)
        except self.SelectError, data:
            apply(fun, ("%s" %self.ERROR,))
            apply(fun, ("%s\n" %data,))
        except:
            apply(fun, ("%s" %self.ERROR,))
```

```
            import traceback
            import StringIO
            tb = StringIO.StringIO()
            traceback.print_exc(file=tb)
            tb.seek(0)
            sys.stderr.write(tb.read())
            if not restricted:
                tb.seek(0)
                apply(fun, ("%s\n" %tb.read(),))
            else:
                apply(fun, ("exception caught\n",))
            tb.close()
        apply(fun, ("%s" %self.ETB,))

    def p_call(self, fun, query, restricted):
        raise self.SelectError, "undefined select"


class UnpackError(Unpack):

    def p_call(self, fun, query, restricted):
        if type(query) == types.StringType:
            raise self.SelectError, query
        else:
            raise self.SelectError, 'query'


class UnpackPyob(Unpack):

    def p_call(self, fun, query, restricted):
        objstr = pickle.dumps(query)
        obj64  = base64.encodestring(zlib.compress(objstr))
        if len(obj64) < len(objstr):
            compressed = 1
        else:
            compressed = 0
            obj64 = objstr   # Achtung: Problem wenn ERROR oder ETB enthalten
        debug("\t\t-> pyob: sending "+'len(obj64)'+" chars\n", 10)
        apply(fun, ("%s\n" %compressed,))
        apply(fun, (obj64,))


class UnpackCount(Unpack):

    def p_call(self, fun, query, restricted):
        try:
            apply(fun, ('len(query)'+'\n',))
        except TypeError:
            raise self.SelectError, "Das Ergebnis %s hat keine Länge." %'query'


class UnpackText(Unpack):

    def p_call(self, fun, query, restricted, pr=1):
        if isinstance(query, region.Region):
            try:
                #str = "<id:%s type:%s>\n" %(query.regionid, query.qdb_type)
                apply(fun, ("<id:%s type:%s>\n" %(query.regionid, query.qdb_type), ))
            except AttributeError:
                try:
                    apply(fun, ("<id:%s>\n" %query.regionid,))
```

```
                except AttributeError:
                    pass
            #query.writeTagzeilen(fun)
            mm = map(None, query.getWoerter(),
                    query.getTags(),
                    query.getLemmata())
            for (w,t,l) in mm:
                apply(fun, ("%s\t%s\t%s\n" %(w,t,l),))
            try:
                delimiter = query.getDelimiter()
            except AttributeError:
                delimiter = "<unbestimmter Regionentyp>\n"
            apply(fun, ("%s" %delimiter,))
        elif type(query) == types.StringType:
            if not pr:              # nur auf oberster Ebene erlaubt
                raise ValueError
            apply(fun, ("%s\n" %query,))
        elif type(query) == types.ListType:
            for i in query:
                self.p_call(fun, i, restricted, 0)
        else:
            if type(query) == types.InstanceType:
                what = query.__class__.__name__
            else:
                what = `type(query)`
            if what == "RegionenSlice":      # warum notwendig?
                try:
                    apply(fun, ("<id:%s>\n" %query.regionid,))
                except AttributeError:
                    pass
                query.writeTagzeilen(fun)
            else:
                debug("\ttrying to output %s\n" %what)
                raise TypeError, "trying to output %s\n" %what
        debug("\t\t-> sent\n", 10)


class UnpackItem(Unpack):                # kombinieren mit IUHelp

    def p_call(self, fun, query, restricted):
        self.p_item(fun, query, restricted)

class UnpackList(Unpack):                # kombinieren mit IUHelp

    def p_call(self, fun, query, restricted):
        #if isinstance(query, region.Regionen):
        #    apply(fun, ("<%s items in %s>\n" %(`len(query)`,`query`),))
        if type(query) == types.ListType:
            apply(fun, ("<%s items in list>\n" %`len(query)`,))
        else:
            self.p_item(fun, query, restricted)
            return
        for i in query:
            self.p_call(fun, i, restricted)

class IUHelp:

    def p_item(fun, query, restricted):
        raise NotImplementedError

class IUHRepr(IUHelp):
```

```
1738
1739        def p_item(self, fun, query, restricted):
1740            apply(fun, ("%s\n" %‘query‘,))
1741
1742    class UnpackRepr(UnpackItem, IUHRepr): pass
1743    class UnpackListRepr(UnpackList, IUHRepr): pass
1744
1745
1746    class IUHRepr80(IUHelp):
1747
1748        def p_item(self, fun, query, restricted):
1749            repr = ‘query‘
1750            line = ‘len(repr)‘ + ": " + repr
1751            apply(fun, ("%s\n" %line[:80],))
1752
1753    class UnpackRepr80(UnpackItem, IUHRepr80): pass
1754    class UnpackListRepr80(UnpackList, IUHRepr80): pass
1755
1756
1757    class IUHDebug(IUHelp):
1758
1759        def p_item(self, fun, query, restricted):
1760            apply(fun, ("type = %s\n" %‘type(query)‘,))
1761            if isinstance(query, region.Region):
1762                apply(fun, ("is instance of region.Region\n",))
1763                apply(fun, (".getWoerter = %s\n" %‘query.getWoerter()‘,))
1764            if isinstance(query, region.Regionen):
1765                apply(fun, ("is instance of region.Regionen\n",))
1766                apply(fun, (".subregionen = %s\n" %‘query.subregionen‘,))
1767            import segmentdb
1768            if isinstance(query, segmentdb.SegmentVonDb):
1769                apply(fun, ("is instance of segmentdb.SegmentVonDb\n",))
1770                apply(fun, (".getWoerter = %s\n" %‘query.getWoerter()‘,))
1771                apply(fun, (".subregionen = %s\n" %‘query.subregionen‘,))
1772                apply(fun, (".regionid = %s\n" %‘query.regionid‘,))
1773            apply(fun, ("dir = %s\n" %‘dir(query)‘,))
1774            for attr in dir(query):
1775                apply(fun, (".%s = %s\n" %(attr, ‘getattr(query, attr)‘),))
1776
1777
1778    class UnpackDebug(UnpackItem, IUHDebug): pass
1779    class UnpackListDebug(UnpackList, IUHDebug): pass
1780
1781
1782    class IUHString(IUHelp):
1783
1784        def p_item(self, fun, query, restricted):
1785            apply(fun, ("%s\n" %str(query),))
1786
1787    class UnpackString(UnpackItem, IUHString): pass
1788    class UnpackListString(UnpackList, IUHString): pass
1789
1790
1791    class IUHString80(IUHelp):
1792
1793        def p_item(self, fun, query, restricted):
1794            s = str(query)
1795            line = ‘len(s)‘ + ": " + s
1796            apply(fun, ("%s\n" %line[:80],))
1797
1798    class UnpackString80(UnpackItem, IUHString80): pass
```

```
1799 class UnpackListString80(UnpackList, IUHString80): pass
1800
1801
1802 class IUHID(IUHelp):
1803
1804     def p_item(self, fun, query, restricted):
1805         apply(fun, ("%s\n" %‘query.regionid‘,))
1806
1807 class UnpackID(UnpackItem, IUHID): pass
1808 class UnpackListID(UnpackList, IUHID): pass
1809
1810
1811 class QueryRequestHandler(SocketServer.BaseRequestHandler):
1812
1813     def __init__(self, request, client_address, server, qrhid, dbconn):
1814         self.id = qrhid
1815         self.dbconn = dbconn
1816         SocketServer.BaseRequestHandler.__init__(self, request,
1817                                                  client_address, server)
1818
1819     def setup(self):
1820         self.query = Query(self.dbconn)
1821         self.query.setCache(self.server.cache)
1822         self.query.setServer(self.server)
1823         self.query.setClientID(self.id)
1824
1825     def handle(self):
1826         try:
1827             if self.query.run(self.request, self.client_address):
1828                 # query will be closed in finish()
1829                 self.server.stop = 1   # stop server
1830         except:
1831             import traceback
1832             import StringIO
1833             tb = StringIO.StringIO()
1834             traceback.print_exc(file=tb)
1835             tb.seek(0)
1836             sys.stderr.write("\twhile handling %s:\n" %‘self.client_address‘)
1837             zeile = tb.readline()
1838             while zeile != "":
1839                 sys.stderr.write("\t"+zeile)
1840                 zeile = tb.readline()
1841
1842     def finish(self):
1843         self.query.close()
1844         self.server.qrhFinishedMutex.acquire()
1845         self.server.qrhFinished[self.id] = 1       # alter server-data
1846         self.server.qrhFinishedMutex.release()
1847
1848
1849 class QueryServer(SocketServer.ThreadingTCPServer, zeitmessung.Zeitmessung):
1850
1851     def __init__(self, port):
1852         zeitmessung.Zeitmessung.__init__(self)
1853         self.zeitMutex = thread.allocate_lock()
1854         self.cache = cache.MutexDefaultCache(self)
1855         #self.cache = cache.NullCache(self)
1856         self.qrhFinished = {}
1857         self.qrhFinishedMutex = thread.allocate_lock()
1858         self.nextqrhid = 0
1859         self.info = {}
```

```
1860            self.infoMutex = thread.allocate_lock()
1861            self.jobs = []
1862            self.jobsMutex = thread.allocate_lock()
1863            self.dbconn = db.Db()
1864            self.tcount = 0
1865            self.tlast = 0
1866            self.ac_tclast = 0        # autoclear-check: letzter tcount
1867            self.ac_tcdist = 60*5     # tcounts bis zum naechsten Check
1868            self.ac_trdist = 60*10    # Zeit bis zum naechsten Check
1869            self.ac_trlast = time.time()
1870            self.ac_mem = config.qdb_ac_mem
1871            self.ac_mem2 = 0
1872            self.start = self.ctime()
1873            self.port  = port
1874            initVersion()
1875            SocketServer.ThreadingTCPServer.__init__(self, ("", port),
1876                                                     QueryRequestHandler)
1877
1878        def process_jobs(self):
1879            self.jobsMutex.acquire()
1880            num_jobs = len(self.jobs)
1881            self.jobsMutex.release()
1882            for index in range(num_jobs):
1883                self.jobsMutex.acquire()
1884                if len(self.jobs):
1885                    job = self.jobs[0]
1886                else:
1887                    job = None
1888                self.jobsMutex.release()
1889                if not job:
1890                    continue
1891                try:
1892                    cid = job[0]
1893                    name = job[1]
1894                    if name == "setCache":
1895                        ncache = job[2]
1896                        debug('self.cache.statistic()'+'\n')
1897                        debug("*** neuer Cache ***", 2)
1898                        self.cache = ncache
1899                except:
1900                    import traceback
1901                    import StringIO
1902                    tb = StringIO.StringIO()
1903                    traceback.print_exc(file=tb)
1904                    tb.seek(0)
1905                    sys.stderr.write("\twhile doing job %s:\n" %'job')
1906                    zeile = tb.readline()
1907                    while zeile != "":
1908                        sys.stderr.write("\t"+zeile)
1909                        zeile = tb.readline()
1910                self.jobsMutex.acquire()
1911                if len(self.jobs):
1912                    del self.jobs[0]
1913                self.jobsMutex.release()
1914
1915        def count_time(self, step):
1916            self.tcount = self.tcount + step
1917            if self.tcount > (60*30):
1918                self.tcount = 0
1919                if time.time() - self.tlast > (60*60):
1920                    self.tlast = time.time()
```

```
            debug('*** %s ***\n' % time.ctime(self.tlast), -5)

    def check4autoclear(self):
        if self.ac_tclast <= self.tcount < (self.ac_tclast + self.ac_tcdist):
            return
        self.ac_tclast = self.tcount
        now = time.time()
        if now < (self.ac_trlast + self.ac_trdist):
            return
        self.ac_trlast = now
        try:
            mem = getVmSize()
            l = string.split(mem)
            zahl    = l[0]
            einheit = l[1]
            mem_kb = int(zahl)
            if einheit in ["KB", "kB"]:
                pass
            elif einheit in ["MB", "mB"]:
                mem_kb = mem_kb * 1024
            else:
                raise Exception
            mem = mem_kb
        except:
            debug("*** autoclear failed: unable to get VmSize ***\n", -12)
            return
        mem2 = self.ac_mem2
        if mem2 < self.ac_mem:           # self.ac_mem nicht unterschreiten
            mem2 = self.ac_mem
        if mem2 >= mem:                  # mem noch unter dem Limit mem2?
            mem2 = self.ac_mem2 / 2      # mem2 verkleinern
            if mem2 < self.ac_mem:       # self.mem nicht unterschreiten
                self.ac_mem2 = self.ac_mem
            else:
                self.ac_mem2 = mem2
            return
        self.ac_mem2 = mem               # aktuellen Speicherbedarf als Limit
        qrhid = self.nextqrhid
        self.nextqrhid = qrhid + 1
        self.qrhFinishedMutex.acquire()
        self.qrhFinished[qrhid] = 0
        self.qrhFinishedMutex.release()
        self.infoMutex.acquire()
        s = "cache autoclear %s startet on %s" %(‘qrhid‘,‘self.ctime()‘)
        self.info[qrhid] = s
        self.infoMutex.release()
        import threading
        t = threading.Thread(target = self.p_autoclear, args = (qrhid, ))
        t.start()
        debug("*** autoclear startet on %s ***\n" %self.ctime(), -5)

    def p_autoclear(self, qrhid):
        self.cache.invalidate()
        self.infoMutex.acquire()
        s = "cache autoclear %s finished on %s" %(‘qrhid‘,‘self.ctime()‘)
        self.info[qrhid] = s
        self.infoMutex.release()
        self.qrhFinishedMutex.acquire()
        self.qrhFinished[qrhid] = 1       # alter server-data
        self.qrhFinishedMutex.release()
```

```
1982     def get_request(self):
1983         r = []
1984         while not self.stop and not r:
1985             r,w,e = select.select([self.socket], [], [], 4)  # wait max 4 sec
1986             self.count_time(4)
1987             self.process_jobs()
1988             self.check4autoclear()
1989         if r:
1990             return self.socket.accept()
1991         else:
1992             return (None, None)
1993 
1994     def verify_request(self, request, client_address):
1995         if request:
1996             return 1
1997         return 0
1998 
1999     def finish_request(self, request, client_address):
2000         qrhid = self.nextqrhid
2001         self.nextqrhid = qrhid + 1
2002         self.qrhFinishedMutex.acquire()
2003         self.qrhFinished[qrhid] = 0
2004         self.qrhFinishedMutex.release()
2005         self.infoMutex.acquire()
2006         self.info[qrhid] = "client %s startet on %s" %(‘qrhid‘,‘self.ctime()‘)
2007         self.infoMutex.release()
2008         dbconn = self.dbconn
2009         self.RequestHandlerClass(request, client_address, self, qrhid, dbconn)
2010 
2011     def handle_error(self, conn, client_address):
2012         self.stop = 1   # called if RequestHandler throws an exception
2013                         # (according to docu, but that’s wrong if threaded)
2014 
2015     def ctime(self):
2016         return time.ctime(time.time())
2017 
2018     def serve_until_shutdown(self):
2019         self.stop = 0
2020         self.tlast = time.time()
2021         debug(’*** Service started on %s ***\n’ % self.ctime(), -7)
2022         while not self.stop:
2023             self.handle_request()
2024             self.qrhFinishedMutex.acquire()
2025             self.del_info_where_finished()
2026             self.del_1_from_qrhfinished()
2027             self.qrhFinishedMutex.release()
2028         debug(’*** Service stopped on %s ***\n’ % self.ctime(), -7)
2029         time.sleep(0.05)
2030         self.qrhFinishedMutex.acquire()
2031         qrh_running = (0 in self.qrhFinished.values())
2032         self.qrhFinishedMutex.release()
2033         while qrh_running:  # wait for termination of query threads
2034             time.sleep(2.0)
2035             self.qrhFinishedMutex.acquire()
2036             self.del_info_where_finished()
2037             self.del_1_from_qrhfinished()
2038             debug(’waiting for childs %s\n’ %‘self.qrhFinished.keys()‘, -6)
2039             qrh_running = (0 in self.qrhFinished.values())
2040             self.qrhFinishedMutex.release()
2041         debug(‘self.cache.statistic()‘+’\n’)
2042 
```

```
2043      def del_info_where_finished(self):
2044          self.infoMutex.acquire()
2045          keys = self.info.keys()
2046          self.infoMutex.release()
2047          for cid in keys:
2048              self.infoMutex.acquire()
2049              if self.info.has_key(cid) \
2050              and self.qrhFinished.has_key(cid) \
2051              and self.qrhFinished[cid] == 1:
2052                  del self.info[cid]
2053              self.infoMutex.release()
2054
2055      def del_1_from_qrhfinished(self):
2056          for key in self.qrhFinished.keys():
2057              if self.qrhFinished[key] == 1:
2058                  del self.qrhFinished[key]
2059
2060      def close(self):
2061          self.socket.close()
2062          self.dbconn.close()
2063
2064      def p_addToStatistic(self, s):
2065          for name in s.keys():
2066              if self.zeit_in.has_key(name):
2067                  bis_jetzt = self.zeit_in[name]
2068              else:
2069                  bis_jetzt = 0.0
2070              try:
2071                  bis_jetzt = bis_jetzt + s[name]
2072                  self.zeit_in[name] = bis_jetzt
2073              except TypeError:
2074                  debug("Fehler: zeit_in (Server) %s\n" %‘(self.zeit_in,s)‘, -3)
2075
2076
2077  def debug(line, level = 8):
2078      if debug_level >= level:
2079          sys.stderr.write(line)
2080
2081  def reverse(s):
2082      m = len(s)-1
2083      r = ""
2084      for i in range(m+1):
2085          r = r + s[m-i]
2086      return r
2087
2088  def getVmSize():
2089      pid = os.getpid()
2090      pinfo = open("/proc/%s/status" %pid, "r")
2091      size = "unbekannt"
2092      if pinfo:
2093          line = pinfo.readline()
2094          while line != "":
2095              if line[:7] == "VmSize:":
2096                  size = string.strip(line[7:])
2097              line = pinfo.readline()
2098          pinfo.close()
2099      return size
2100
2101  def initVersion():
2102      global version
2103      version = {}
```

```
    if os.path.exists(config.databaseapi + "CVS/Entries"):
        entries = open(config.databaseapi + "CVS/Entries", "r")
        line = entries.readline()
        while line != "":
            d = string.split(line, '/')
            if len(d) >= 3 and d[1] in ["querydb.py", "config.py", "db.py"]:
                version[d[1]] = d[2]
            line = entries.readline()
        entries.close()

if __name__ == '__main__':
    sys.stderr.write("Please use ../demo-qdb.py\n")
```

qclient.py

Der Datenbankclient.

```
#!/usr/bin/python

# qclient.py: Client zum querydb-Server

import secret
import cache
import config
import regiondb
from socket import *
import base64
import pickle
import string
import sys
import zlib

class Query: #{

  def __init__(self, address = None):
      if not regiondb.defaultDbconn:
          sys.stderr.write("erst regiondb.setDefaultDbconn() aufrufen!\n")
          raise Exception("erst regiondb.setDefaultDbconn() aufrufen!")
      self.cache = cache.DefaultCache(self)
      self.sock = None
      self.param = {}
      self.paramHash = ""
      if not address:
          for pport in config.qdb_ports:
              if self.connect((config.qdb_host, pport)):
                  address = (config.qdb_host, pport)
                  #self.close()
                  break
      else:
          self.connect(address)
      self.address = address
      sys.stderr.write("Benutze Server " + `address` + "\n")
      self.korplist = None

  def ok(self):
      return self.address

  def connect(self, address = None):
      if self.sock:
          self.close()
```

```
        if not address:
            address = self.address
        if not address:
            return 0
        sock = socket(AF_INET, SOCK_STREAM)
        try:
            sock.connect(address)
        except error, data:
            return 0
        frage = sock.recv(16384)
        pos = string.find(frage, 'getValue(')
        start = string.find(frage, "'", pos) + 1
        ende  = string.find(frage, "'", start)
        if (start >= 0) and (ende >= 0) and (start <= ende):
            key = frage[start:ende]
        else:
            sock.close()
            return 0
        sock.send(secret.getValue(key))
        antwort = sock.recv(16384)
        if string.rstrip(antwort) == "OK":
            self.sock = sock
            return 1
        else:
            sock.close()
            return 0

        # key aus Frage extrahieren, getValue aufrufen, Antwort verschicken

    def close(self):
        if self.sock:
            self.sock.close()
            self.sock = None

    def p_checkNotInStr(self, badchars, str):
        for c in badchars:
            if c in str:
                raise ValueError  # gibt's dafuer eine besser passende Exception?

    def p_updateParam(self, key, value):
        self.param[key] = value
        liste = self.param.values()
        liste.sort()
        self.paramHash = 'liste'

    def p_setcorpus(self, idlststr):
        self.p_checkNotInStr(';\n', idlststr)
        if self.sock:
            self.p_process_line("u:" + idlststr)
        self.p_updateParam("corpus", "u:" + idlststr)

    def setkorpus(self, qdb_type):
        self.p_checkNotInStr(':;\n', qdb_type)
        self.p_setcorpus(qdb_type)

    def setwbkorpus(self):
        self.p_setcorpus("wb")

    def seterkwbkorpus(self):
        self.p_setcorpus("wb+erkannt")
```

```
105     def seterkkorpus(self):
106         self.p_setcorpus("erkannt")
107 
108     def p_setParam(self):
109         for line in self.param.values():
110             self.p_process_line(line)
111 
112     def p_getAntwort(self):
113         antwort = self.sock.recv(16384)
114         ETB = "<DONE>\n"          # auf Wunsch von Arno ein verstaedlicher String
115         linesrecved = string.count(antwort, "\n")
116         fertig = (string.find(antwort, ETB) >= 0)
117         while not fertig:
118             temp = self.sock.recv(16384)
119             if not temp:                # connection closed?
120                 raise Exception, "connection closed before ETB"
121             linesrecved = linesrecved + string.count(temp, "\n")
122             antwort = antwort + temp
123             if len(temp) > len(ETB):
124                 fertig = (string.find(temp, ETB) >= 0)     # schnell
125             else:
126                 fertig = (string.find(antwort, ETB) >= 0)  # langsam
127         return antwort[:string.find(antwort, ETB)]
128 
129     def p_isCommand(self, line):
130         if line[:8] in ["shutdown"]:
131             return 1
132         if line[:6] in ["status"]:
133             return 1
134         if line[:5] in ["cache", "debug"]:
135             return 1
136         if line[:4] in ["info"]:
137             return 1
138         if line[:2] in ["u:"]:
139             return 1
140         return 0
141 
142     def p_isUpdate(self, line):
143         if string.find(line,"setparse") >= 0:
144             return 1
145         else:
146             return 0
147 
148     def p_process_line(self, line):
149         isUpdate = self.p_isUpdate(line)
150         if self.cache.has_key((self.paramHash, line)) \
151         and not self.p_isCommand(line):            # Kommandos nicht cachen
152             antwort = self.cache[(self.paramHash, line)]
153         elif not self.sock: #{
154             self.connect()
155             if self.param:
156                 self.p_setParam()
157             self.sock.send(line+"\n")
158             antwort = self.p_getAntwort()
159             self.close()
160         #}
161         else:
162             self.sock.send(line+"\n")
163             antwort = self.p_getAntwort()
164         if not isUpdate:
165             self.cache[(self.paramHash, line)] = antwort
```

```
166         else:
167             self.cache.invalidate()
168         return antwort
169 
170     def wort(self, **where):
171         return self.p_execute('wort', where)
172 
173     def lemma(self, **where):
174         return self.p_execute('grundform', where)
175 
176     def satz(self, **where):
177         return self.p_execute('satz', where)
178 
179     def segment(self, **where):
180         return self.p_execute('segment', where)
181 
182     def p_execute(self, name, where):
183         try:
184             self.p_checkNotInStr(';\n', name)
185             q = "t:" + name + ";s:pyob;w:"
186             if len(where.keys()) < 1:
187                 sys.stderr.write("qclient.Query.p_execute: where-Teil leer\n")
188             for key in where.keys():
189                 value = where[key]
190                 self.p_checkNotInStr(';,\n=', key)
191                 self.p_checkNotInStr(';,\n' , value)
192                 q = q + key + "=" + value + ","
193             obj64 = self.p_process_line(q[:-1])    # letztes Komma wegschneiden
194             if obj64[:8] == "<ERROR>\n":
195                 sys.stderr.write(obj64)
196                 raise Exception
197             #sys.stderr.write("obj64: ### BEGIN ###\n" + obj64 + "### END ###\n")
198             if obj64[:2] == "1\n":
199                 objstr = base64.decodestring(obj64[2:])
200                 objstr = zlib.decompress(objstr)
201             else:
202                 objstr = obj64[2:]
203             return pickle.loads(objstr)
204         except:
205             return [] #kann passieren, wenn Satzzeichen dazwischenkommen :(
206 
207     def query(self, line):
208         return (self.p_process_line(string.rstrip(line)), None)
209 
210     def unpack(self, writefun, object, select):
211         apply(writefun, (object,))
212 
213 #}
214 
215 def main():
216     regiondb.setDefaultDbconn(db.Db())
217     qdb = None
218     try:
219         if len(sys.argv) > 1:
220             port = sys.argv[1]
221             qdb = Query((config.qdb_host, int(port)))
222         else:
223             qdb = Query()
224         while 1:
225             sys.stdout.write("->")
226             line = raw_input()
```

```
            if not line:
                break
            #            if qdb.connect():
            query, select = qdb.query(line)
            qdb.unpack(sys.stdout.write, query, select)
        qdb.close()
        sys.exit()
    except error, data:
        print error, data
        if qdb:
            qdb.close()
        sys.exit(1)

if __name__ == '__main__':
    main()

```

cache.py

Ein Cache, zum Speichern auf die Platte.

```
#/usr/bin/python

# Modul Lexikon.cache

import sys
import thread

class Cache:

    def has_key(self, key):
        self.miss = self.miss + 1
        return 0

    def keys(self):
        return []

    def values(self):     # list of values
        liste = []
        for key in self.keys():
            liste.append(self[key])
        return liste

    def items(self):     # list of (key, value) pairs
        liste = []
        for key in self.keys():
            liste.append((key, self[key]))
        return liste

    def __getitem__(self, key):
        self.hit = self.hit + 1
        return None

    def __setitem__(self, key, value):
        self.write = self.write + 1
        pass

    def __delitem__(self, key):
        self.dele = self.dele + 1
        pass

    def __init__(self, dummy):
```

```
42          self.initstat()
43
44      def initstat(self):
45          self.miss = 0
46          self.hit = 0
47          self.write = 0
48          self.dele = 0
49
50      def __repr__(self):
51          return ‘{}‘
52
53      def __getstate__(self):
54          return self.items()
55
56      def __setstate__(self, state):
57          self.initstat()
58
59      def invalidate(self):
60          for key in self.keys():
61              del self[key]
62
63      def statistic(self):
64          total = self.hit+self.miss
65          if total > 0:
66              saved = 100*self.hit/total
67          else:
68              saved = 0
69          return {"miss"  : self.miss,
70                  "hit"   : self.hit,
71                  "reads saved in %" : saved,
72                  "write" : self.write,
73                  "del"   : self.dele}
74
75  class NullCache(Cache):
76      pass
77
78  class FullCache(Cache):
79
80      def has_key(self, key):
81          retval = self.inhalt.has_key(key)
82          if not retval:
83              self.miss = self.miss + 1
84          return retval
85
86      def keys(self):
87          Cache.keys(self)
88          return self.inhalt.keys()
89
90      def __getitem__(self, key):
91          Cache.__getitem__(self, key)
92          return self.inhalt[key]
93
94      def __setitem__(self, key, value):
95          Cache.__setitem__(self, key, value)
96          self.inhalt[key] = value
97
98      def __delitem__(self, key):
99          Cache.__delitem__(self, key)
100         del self.inhalt[key]
101
102     def __init__(self, basis):
```

```
103         Cache.__init__(self, basis)
104         self.inhalt = {}
105 
106     def __setstate__(self, state):
107         Cache.__setstate__(self, [])
108         self.inhalt = {}
109         for (key, value) in state:
110             self[key] = value
111 
112     def __repr__(self):
113 #        Cache.(self)
114         return 'self.inhalt'
115 
116 class MutexDefaultCache(FullCache):
117 
118     def has_key(self, key):       # Warnung: Zustand kann sich aendern
119         self.mutex.acquire()
120         retval = FullCache.has_key(self, key)
121         self.mutex.release()
122         return retval
123 
124     def keys(self):               # Warnung: Zustand kann sich aendern
125         self.mutex.acquire()
126         retval = FullCache.keys(self)
127         self.mutex.release()
128         return retval
129 
130     def items(self):     # list of (key, value) pairs
131         liste = []
132         for key in self.keys():
133             try:                            #  sicherstellen, dass Paar noch da
134                 item = (key, self[key])
135                 liste.append(item)
136             except KeyError:
137                 pass
138         return liste
139 
140     def __getitem__(self, key):
141         self.mutex.acquire()
142         keyerror = 0
143         try:
144             retval = FullCache.__getitem__(self, key)
145         except KeyError:
146             self.miss = self.miss + 1
147             keyerror = 1
148         self.mutex.release()
149         if keyerror:
150             raise KeyError
151         return retval
152 
153     def __setitem__(self, key, value):
154         self.mutex.acquire()
155         FullCache.__setitem__(self, key, value)
156         self.mutex.release()
157 
158     def __delitem__(self, key):
159         self.mutex.acquire()
160         FullCache.__delitem__(self, key)
161         self.mutex.release()
162 
163     def __init__(self, basis):
```

```
164         FullCache.__init__(self, basis)
165         self.mutex = thread.allocate_lock()
166 
167     def __setstate__(self, state):
168         FullCache.__setstate__(self, [])
169         self.mutex = thread.allocate_lock()
170         for (key, value) in state:
171             self[key] = value
172 
173 
174 class SkipCache(Cache):
175 
176     def has_key(self, key):
177         Cache.has_key(self, key)
178         return self.inhalt.has_key(key)
179 
180     def keys(self):
181         Cache.keys(self)
182         return self.inhalt.keys()
183 
184     def __getitem__(self, key):
185         Cache.__getitem__(self, key)
186         return self.inhalt[key]
187 
188     def __setitem__(self, key, value):
189         if len(self.keys()) > self.limit:       # 2do: Speicherbelegung pruefen
190           self.inhalt = {}
191         Cache.__setitem__(self, key, value)
192         self.inhalt[key] = value
193 
194     def __delitem__(self, key):
195         Cache.__delitem__(self, key)
196         del self.inhalt[key]
197 
198     def __init__(self, basis):
199         Cache.__init__(self, basis)
200         self.inhalt = {}
201 
202     def __repr__(self):
203 #         Cache.(self)
204         return 'self.inhalt'
205 
206 
207 class OrderedCache(Cache): #{                # abstrakte Klasse
208 
209     def p_newStateFor(self, key, value):     # ueberschreiben
210         return 0
211 
212     def p_stateCmp(self, state1, state2):    # ueberschreiben
213         return 0
214 
215     def has_key(self, key):
216         Cache.has_key(self, key)
217         return self.inhalt.has_key(key)
218 
219     def keys(self):
220         Cache.keys(self)
221         return self.inhalt.keys()
222 
223     def __getitem__(self, key):
224         Cache.__getitem__(self, key)
```

```
225         value, index = self.inhalt[key]
226         state, keyv = self.heap[index]
227         if key != keyv:
228             sys.stderr.write("Fehler in DatabaseAPI.cache.OrderedCache\n")
229         state = self.p_newStateFor(state, key, value)
230         self.heap[index] = (state, key)
231         # 2do: Kosten anpassen und del self.heap[0] aufrufen, bis gering genug
232         # !!! erstmal hier nicht weiter machen, da SkipCache zuerst schreiben
233         return value
234 
235     def __setitem__(self, key, value):
236         Cache.__setitem__(self, key, value)
237         # 2do
238         self.inhalt[key] = value
239 
240     def __delitem__(self, key):
241         Cache.__delitem__(self, key)
242         # 2do
243         del self.inhalt[key]
244 
245     def __init__(self, basis):
246         Cache.__init__(self, basis)
247         # 2do
248         self.inhalt = {}
249 
250     def __repr__(self):
251 #         Cache.(self)
252         # 2do
253         return 'self.inhalt'
254 #}
255 
256 
257 class Heap: #{
258 
259     def __init__(self, cmpF, size):
260         self.cmpF = cmpF
261         self.data = size * [None]
262 
263     #2do : setitem -> Reorganisation
264 
265 #}
266 
267 
268 class DefaultCache(FullCache):
269     pass                        # 2do: Methoden def.
270 
271 
272 class Cached(DefaultCache): #{            # Cache durch Vererbung
273 
274     def loaditem(self, key):              # ueberschreiben
275         return None
276 
277     def saveitem(self, key, value):       # ueberschreiben
278         pass
279 
280     def loadable(self, key):              # ueberschreiben
281         return 0
282 
283     def __init__(self, client):
284         DefaultCache.__init__(self, client)
285 
```

```
    def __getitem__(self, key):
        if DefaultCache.has_key(self, key):
          return DefaultCache.__getitem__(self, key)
        value = self.loaditem(key)
        DefaultCache.__setitem__(self, key, value)
        return value

    def __setitem__(self, key, value):
        self.saveitem(key, value)
        DefaultCache.__setitem__(self, key, value)

    def haskey(self, key):
        return DefaultCache.has_key(self, key) or \
               self.loadable(key)
#}

if __name__ == '__main__':
    mc = MutexDefaultCache(None)
    mc['a'] = 'A'
    mc['b'] = 'B'
    print mc['a']
    mc['c'] = 'C'
    print mc['b']
    print mc.statistic()
```

config.py

Konfigurationseinzelheiten.

```
#!/usr/bin/python

# Modul config.py
#
# Attribute:
#    dbname
#    hostname
#    user
#    tagger_de
#    tagger_en
#    wbdateien

import getpass
import os
import popen2
import string
import sys

config = {

#
# Name und Typ der Datenbank
#

"dbname"    : {"koks@nulix"        : "mogel17b" ,
               "bkoch@nulix"       : "alternativ" ,
               "logotax@nulix"     : "neu17" ,
               "jowagner@nulix"    : "neu17" ,
               "ptschorn@nulix"    : "koks" ,
               "aerpenbe@nulix"    : "koks" ,
               "britta@inanna.surf2000.de"           : "neu",
```

```
                "koks@inanna.surf2000.de"           : "wb",
                "koks@gorina"      : "neu17b" ,
               },
"dbtype"    : {"koks@nulix"        : "mysql" ,
                "bkoch@nulix"       : "mysql" ,
                "logotax@nulix"     : "mysql" ,
                "jowagner@nulix"    : "mysql" ,
                "ptschorn@nulix"    : "mysql" ,
                "aerpenbe@nulix"    : "mysql" ,
                "koks@gorina"      : "mysql"   ,
                "britta@inanna.surf2000.de" : "mysql",
                "koks@inanna.surf2000.de" : "mysql",
               },
"dbuser"    : {"koks@nulix"        : "koks" ,
                "bkoch@nulix"       : "bkoch" ,
                "logotax@nulix"     : "koks" ,
                "jowagner@nulix"    : "koks" ,
                "ptschorn@nulix"    : "koks" ,
                "aerpenbe@nulix"    : "koks" ,
                "britta@inanna.surf2000.de" : "britta",
                "koks@inanna.surf2000.de" : "koks",
                "koks@gorina"      : "koks" ,
               },
"dbpass"    : {"koks@nulix"        : "iKK4l0mn" ,
                "bkoch@nulix"       : "onK75Tdo",
                "logotax@nulix"     : "iKK4l0mn",
                "jowagner@nulix"    : "iKK4l0mn",
                "ptschorn@nulix"    : "iKK4l0mn",
                "aerpenbe@nulix"    : "iKK4l0mn",
                "britta@inanna.surf2000.de" : "iKK4l0mn",
                "koks@inanna.surf2000.de" : "iKK4l0mn",
                "koks@gorina"      : "iKK4l0mn",
               },

#
# querydb Server, den der Client benutzen soll
#

"qdb_ports" : {"koks@nulix"        : [58597,45628,29295,55776,46581,55149],
                "bkoch@nulix"       : [58597,45628,29295,55776,46581,55149],
                "logotax@nulix"     : [55776,55149],
                "jowagner@nulix"    : [51111,45628,29295,55776,46581,55149],
                "ptschorn@nulix"    : [58597,45628,29295,55776,46581,55149],
                "aerpenbe@nulix"    : [58597,45628,29295,55776,46581,55149],
                "koks@gorina"      : [58597,45628,29295,55776,46581,55149],
                "britta@inanna.surf2000.de" : [58597,45628,29295,55776,46581,55149],
                "koks@inanna.surf2000.de" : [58597,45628,29295,55776,46581,55149],
               },

"qdb_host"  : {"koks@nulix"        : "nulix.cl-ki.uni-osnabrueck.de",
                "bkoch@nulix"       : "nulix.cl-ki.uni-osnabrueck.de",
                "logotax@nulix"     : "nulix.cl-ki.uni-osnabrueck.de",
                "jowagner@nulix"    : "nulix.cl-ki.uni-osnabrueck.de",
                "ptschorn@nulix"    : "nulix.cl-ki.uni-osnabrueck.de",
                "aerpenbe@nulix"    : "nulix.cl-ki.uni-osnabrueck.de",
                "koks@gorina"      : "localhost",
                "britta@inanna.surf2000.de" : "localhost",
                "koks@inanna.surf2000.de" : "localhost",
               },

#
```

```
93  # querydb Server cache.autoclear Schwelle
94  #
95
96  "qdb_ac_mem" : {"koks@nulix"       : 102102,
97                  "bkoch@nulix"      : 102102,
98                  "logotax@nulix"    : 102102,
99                  "jowagner@nulix"   : 102102,
100                 "ptschorn@nulix"   : 102102,
101                 "aerpenbe@nulix"   : 102102,
102                 "koks@gorina"      : 204204,
103                 "britta@inanna.surf2000.de" : 102102,
104                 "koks@inanna.surf2000.de" : 102102,
105                },
106
107 #
108 # Pfad zum Aligner
109 #
110
111 "align"     : {"koks@nulix"        : "/home/koks/align/align.sh" ,
112                "bkoch@nulix"       : "./align" ,
113                "jowagner@nulix"    : "/home/jowagner/bin/align" ,
114                "logotax@nulix"     : "/home/logotax/bin/align" ,
115                "aerpenbe@nulix"    : "/home/koks/align/align.sh" ,
116                "ptschorn@nulix"    : "./align" ,
117                "koks@gorina"      : "/home/koks/align/align.sh" ,
118                "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/align.sh" ,
119                "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/align.sh" ,
120               },
121
122 "al_jar"    : {"koks@nulix"        : "/home/koks/align/al.jar" ,
123                "bkoch@nulix"       : "/home/bkoch/koks/align/al.jar" ,
124                "jowagner@nulix"    : "/home/jowagner/align/al.jar" ,
125                "logotax@nulix"     : "/home/logotax/align/al.jar" ,
126                "aerpenbe@nulix"    : "./al.jar" ,
127                "ptschorn@nulix"    : "./al.jar" ,
128                "koks@gorina"      : "/home/koks/align/al.jar" ,
129                "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/al.jar" ,
130                "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/al.jar" ,
131               },
132
133 #
134 # Tagfolgenverzeichnis
135 #
136 "tags"      : {"koks@nulix"        : "/home/koks/align/tags" ,
137                "bkoch@nulix"       : "tags" ,
138                "jowagner@nulix"    : "/home/jowagner/align/tags" ,
139                "logotax@nulix"     : "/home/logotax/align/tags" ,
140                "aerpenbe@nulix"    : "./tags" ,
141                "ptschorn@nulix"    : "./tags" ,
142                "koks@gorina"      : "/home/koks/align/tags" ,
143                "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/tags" ,
144                "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/tags" ,
145               },
146
147 #
148 # DatabaseAPI-Verzeichnis
149 #
150 "databaseapi": {"koks@nulix"       : "/home/koks/align/DatabaseAPI/" ,
151                 "bkoch@nulix"      : "./DatabaseAPI/" ,
152                 "jowagner@nulix"   : "/home/jowagner/align/DatabaseAPI/" ,
153                 "logotax@nulix"    : "/home/logotax/align/DatabaseAPI/" ,
```

```
                "aerpenbe@nulix"    : "./DatabaseAPI/" ,
                "ptschorn@nulix"    : "./DatabaseAPI/" ,
                "koks@gorina"      : "/home/koks/align/DatabaseAPI/" ,
                "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/DatabaseAPI/" ,
                "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/DatabaseAPI/" ,
               },

#
# Pfad zum IMS Tree-Tagger
#

"tagger_de" : {"koks@nulix"        : "/home/koks/bin/tree-tagger-german" ,
                "bkoch@nulix"       : "/home/bkoch/koks/bin/tree-tagger-german" ,
                "jowagner@nulix"    : "/home/jowagner/bin/tree-tagger-german" ,
                "logotax@nulix"     : "/home/logotax/bin/tree-tagger-german" ,
                "aerpenbe@nulix"    : "/home/aerpenbe/bin/tree-tagger-german" ,
                "ptschorn@nulix"    : "/home/koks/bin/tree-tagger-german" ,
                "koks@gorina"      : "/home/koks/bin/tree-tagger-german" ,
                "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-german",
                "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-german",
               },
"tagger_en" : {"koks@nulix"        : "/home/koks/bin/tree-tagger-english" ,
                "bkoch@nulix"       : "/home/bkoch/koks/bin/tree-tagger-english",
                "jowagner@nulix"    : "/home/jowagner/bin/tree-tagger-english" ,
                "logotax@nulix"     : "/home/logotax/bin/tree-tagger-english" ,
                "aerpenbe@nulix"    : "/home/koks/bin/tree-tagger-english" ,
                "ptschorn@nulix"    : "/home/koks/bin/tree-tagger-english" ,
                "koks@gorina"      : "/home/koks/bin/tree-tagger-english" ,
                "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-english",
                "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-english",
               },

#
# Groesse des zu verwendenen Woerterbuchs, beeinflusst wbdateien
#

"wbsize"    : {"koks@nulix"        : "big",
                "bkoch@nulix"       : "big",
                "jowagner@nulix"    : "small",
                "logotax@nulix"     : "small",
                "aerpenbe@nulix"    : "small",
                "ptschorn@nulix"    : "small",
                "koks@gorina"      : "tiny",
                "britta@inanna.surf2000.de" : "big"  ,
                "koks@inanna.surf2000.de" : "big"  },

#Ids der Woerterbuecher
"wbids": {"koks@nulix"        : "(0,1,2,3,4)",
          "bkoch@nulix"       : "(1,2,1225,1226,1227)",
          "jowagner@nulix"    : "(0,1,2,3,4)",
          "logotax@nulix"     : "(0,1,2,3,4)",
          "ptschorn@nulix"    : "(1,2,3,1160,1161)",
          "aerpenbe@nulix"    : "(1,2,3,1160,1161)",
          "koks@gorina"      : "(0,1,2,3,4)",
          "britta@inanna.surf2000.de" : "(0, 1, 2, 3, 4)",
          "koks@inanna.surf2000.de" : "(0, 1, 2, 3, 4)" },

"goodcorpora": {"koks@nulix"        : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32,
                  "bkoch@nulix"       : "(7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
                 "logotax@nulix"    : "(3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
                 "jowagner@nulix"    : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32,
```

```
                "ptschorn@nulix"    : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32, 33,
                "aerpenbe@nulix"    : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32, 33,
                "koks@gorina"      : "(10)",
                "britta@inanna.surf2000.de" : "(106, 107, 108, 109, 110, 111,112,113,114)",
                "koks@inanna.surf2000.de" : "(106, 107, 108, 109, 110, 111,112,113,114)",
                }
}

#
# --- ab hier braucht i.d.R. nichts geaendert werden ---
#

lang2id = {"de": '1', "en": '2'}  # Bedeutung der SprachIDs

p_sys = sys.path[0]
if p_sys[:1] == "/":        # Pfad absolut?
    p_cwd2 = p_sys          # Ja -> uebernehmen
else:
    p_cwd = os.getcwd()     # Nein -> hoffentlich akt. Verz. noch unveraendert
    os.chdir(p_cwd + "/%s" %(sys.path[0]))
    p_cwd2 = os.getcwd()
    os.chdir(p_cwd)
is_in_bin = 0
if p_cwd2[-4:] == "/bin":
    is_in_bin = 1
if p_cwd2[-5:] == "/bin/":
    is_in_bin = 1

if is_in_bin and os.environ.has_key("KOKS_BIN_USER"):
    user = os.environ["KOKS_BIN_USER"]
elif os.environ.has_key("KOKS_USER"):
    user = os.environ["KOKS_USER"]
else:
    user = getpass.getuser()
r,w=popen2.popen2("hostname")
w.close()
zeile=r.readline()
hostname = string.strip(zeile)
while zeile != "":
  zeile=r.readline()
r.close()

# Folgende Variablen werden beim Import dieses Modules gesetzt:
dbname  = config["dbname"][user + "@" + hostname]
dbtype  = config["dbtype"][user + "@" + hostname]
dbpass  = config["dbpass"][user + "@" + hostname]
dbuser = config["dbuser"][user + "@" + hostname]
qdb_ports = config["qdb_ports"][user + "@" + hostname]
qdb_host  = config["qdb_host"][user + "@" + hostname]
qdb_ac_mem  = config["qdb_ac_mem"][user + "@" + hostname]
align     = config["align"][user + "@" + hostname]
al_jar    = config["al_jar"][user + "@" + hostname]
tags      = config["tags"][user + "@" + hostname]
databaseapi = config["databaseapi"][user + "@" + hostname]
tagger_de = config["tagger_de"][user + "@" + hostname]
tagger_en = config["tagger_en"][user + "@" + hostname]

wbsize = config["wbsize"][user + "@" + hostname]
wbids = config["wbids"][user + "@" + hostname]
goodcorpora = config["goodcorpora"][user + "@" + hostname]
```

```
if wbsize == "tiny": #{
  wbdateien = [ {"de" : "../wb/wbtag/wb2/sk5000de.tag",
                 "en" : "../wb/wbtag/wb2/sk5000en.tag"} ]
#}
elif wbsize == "small": #{
  wbdateien = [ {"de" : "../wb/wbtag/wb2/sk20000de.tag",
                 "en" : "../wb/wbtag/wb2/sk20000en.tag"} ]
#}
elif wbsize == "big": #{
  wbdateien = [ {"de" : "../wb/wbtag/wb2/KOKSdt_eng2ger.tag",
                 "en" : "../wb/wbtag/wb2/KOKSeng_eng2ger.tag"} ]
#}
elif wbsize == "huge": #{
  wbdateien = [ {"de" : "../wb/wbtag/wb2/KOKSdt_eng2ger.tag",
                 "en" : "../wb/wbtag/wb2/KOKSeng_eng2ger.tag"} ,
                {"de" : "../wb/wbtag/wb3/KOKSdt_German.tag",
                 "en" : "../wb/wbtag/wb3/KOKSeng_German.tag"} ,
                {"de" : "../wb/wbtag/wb1/KOKSdt_ger-eng.tag",
                 "en" : "../wb/wbtag/wb1/KOKSeng_ger-eng.tag"} ]
else: #{
  sys.stderr.write("Fehler in config.py: wbsize falsch konfiguriert\n")
  wbdateien = []
#}

def test():
    usersAndHosts = {}
    for key in config.keys():                    # Benutzer sammeln
        for userAndHost in config[key].keys():
            usersAndHosts[userAndHost] = 1
    for key in config.keys():                    # Vollstaendigkeit pruefen
        for userAndHost in usersAndHosts.keys():
            if not config[key].has_key(userAndHost):
                print "In", key, "fehlt", userAndHost

if __name__ == '__main__':
    test()
```

db.py

Eine Hilfsklasse für Datenbankzugriff.

```
#!/usr/bin/python

# Modul DatabaseAPI.db

#TODO: nur postgres!!!
#verkapselt Datenbank-Krams

#import pg
import config
import sys
import thread
import types

p_dbconn_count = 0

class Db:

    def __init__(self, dbtype=config.dbtype, dbname=config.dbname):
        self.initDb(dbtype, dbname)

    def initDb(self, dbtype, dbname):
```

```
22          global p_dbconn_count
23          p_dbconn_count = p_dbconn_count + 1
24          if p_dbconn_count > 10:
25              sys.stderr.write("Warnung: %s db.Db Objekte\n" %p_dbconn_count)
26          if p_dbconn_count > 100:
27              sys.stderr.write("Das sind zu viele. stack trace folgt.\n")
28              import traceback
29              import StringIO
30              tb = StringIO.StringIO()
31              traceback.print_stack(file=tb)
32              tb.seek(0)
33              sys.stderr.write("in %s:\n" %sys.argv[0])
34              zeile = tb.readline()
35              while zeile != "":
36                  sys.stderr.write("\t"+zeile)
37                  zeile = tb.readline()
38          self.dbtype = dbtype
39          self.dbname = dbname
40          if self.dbtype == "pg":
41              import pg
42              self.dbconn = pg.connect(dbname)
43              self.api = 0
44          elif self.dbtype == "mysql":
45              import MySQLdb
46              self.dbconn = MySQLdb.connect(db=dbname, user=config.dbuser, passwd=config.dbpass)
47              self.api = 1
48              #        elif self.dbtype = "oracle":
49              #            import DCOracle
50              #            self.dbconn = DCOracle.Connect("westlotto/fee@q")
51              #            self.api = 1
52          else:
53              sys.stderr.write("dbtype " + ‘dbtype‘ + " unknown\n")
54          self.mutex = thread.allocate_lock()
55
56      def getMaxIndex(self, spalte, tabelle):
57          # 2do: what if two threads append items at the same time?
58          idx = self.executelist("select max(%s) from %s" %(spalte, tabelle))
59          if idx and idx[0] and idx[0][0]:
60              idx = idx[0][0]
61          else:
62              idx = 0L
63          return self.incr(idx)
64
65      def __getstate__(self):
66          return (self.dbtype, self.dbname)
67
68      def __setstate__(self, state):
69          self.initDb(state[0], state[1])
70
71      def incr(self, idx):
72          return unknown2string(long(idx) + 1)
73
74      def executedict(self, query):
75          self.mutex.acquire()
76          #sys.stderr.write("Db.executedict("+‘query‘+")\n")
77          if self.api:
78              self.cursor = self.dbconn.cursor()
79              self.cursor.execute(query)
80              try:
81                  rows = self.cursor.fetchall()
82                  colnames = []
```

```
83                 result = []
84                 for col in cursor.description():
85                     colnames.append(col[0])
86                 for row in rows:
87                     dict = {}
88                     i = 0
89                     for col in row:
90                         dict[colnames[i]] = col
91                         i = i + 1
92                     result.append(dict)
93                 self.mutex.release()
94                 return result
95             except Error, info:
96                 print info
97             else:
98                 self.mutex.release()
99                 raise Exception
100        else:
101            self.results = self.dbconn.query(query)
102            if self.results:
103                retval = self.results.dictresult()
104                self.mutex.release()
105                return retval
106        self.mutex.release()
107        return None
108
109    def executelist(self, query):
110        self.mutex.acquire()
111        #sys.stderr.write("Db.executelist("+‘query‘+")\n")
112        if self.api:
113            self.cursor = self.dbconn.cursor()
114            self.cursor.execute(query)
115            try:
116                retval = self.cursor.fetchall()
117                self.mutex.release()
118                return retval
119            except Error, info:
120                print info
121            else:
122                self.mutex.release()
123                raise Exception
124        else:
125            self.results = self.dbconn.query(query)
126            if self.results:
127                retval = self.results.getresult()
128                self.mutex.release()
129                return retval
130        self.mutex.release()
131        return None
132
133    def update(self, query):
134        self.mutex.acquire()
135        #sys.stderr.write("Db.update("+‘query‘+")\n")
136        if self.api:
137            self.cursor = self.dbconn.cursor()
138            self.cursor.execute(query)
139        else:
140            self.dbconn.query(query)
141        self.mutex.release()
142
143    def close(self):
```

```
144         self.mutex.acquire()
145         if self.dbconn:
146             self.dbconn.close()
147             global p_dbconn_count
148             p_dbconn_count = p_dbconn_count - 1
149             if p_dbconn_count < 0:
150                 sys.stderr.write("Fehler: db.p_dbconn_count = ")
151                 sys.stderr.write("%s\n" %p_dbconn_count)
152         else:
153             sys.stderr.write("Db: None.close()\n")
154         self.dbconn = None        # we should test for it above
155         self.mutex.release()
156 
157     def commit(self):
158         pass
159 
160     def reset(self):
161         self.mutex.acquire()
162         if self.api:
163             self.dbconn.rollback()
164         else:
165             self.dbconn.reset()
166         self.mutex.release()
167 
168 
169 def incr(idx):
170     return unknown2string(long(idx) + 1)
171 
172 def unknown2string(unknown):
173     if type(unknown) == types.StringType:
174         return unknown
175     if type(unknown) == types.LongType:
176         return `unknown`[:-1]
177     if type(unknown) == types.IntType:
178         return `unknown`
179     raise TypeError
180 
```

secret.py

Ein Klasse zum Verschlüsseln von Kommunikation.

```
1  #!/usr/bin/python
2  
3  # Modul secret.py
4  
5  import base64
6  import math
7  import md5
8  import random
9  import string
10 import sys
11 import time
12 
13 def getKey(): #{
14   global salt
15   m = md5.new()
16   m.update(salt)
17   m.update(randomString(128))
18   m.update(`time.time()`)
19   return string.rstrip(base64.encodestring(m.digest()))
20 #}
```

```

def getValue(key): #{
  global salt
  global secret
  m = md5.new()
  m.update(salt)
  m.update(secret)
  m.update(key)
  return string.rstrip(base64.encodestring(m.digest()))

def valueOK(key, value): #{
  return value == getValue(key)
#}


# -- folgende Funktionen und Variablen sind privat --

salt   = "u/v8iI5IB7VOoojnhtLsI7R4a/1GCLqE"
secret = "Yi2UV9C648hByNkVyWTcIfWLDsIQIDY2"

def randomString(bits): #{
  zeichen = bits * math.log(2) / math.log(62) # base64 ohne /+
  tripel = zeichen / 4.0
  retval = ""
  while tripel > 0.0: #{
    s = ""
    for i in range(3):
      char = random.randint(0,255);
      s = s + chr(char)
    id = string.rstrip(base64.encodestring(s))
    if  id == string.replace(id, "/", "")  \
    and id == string.replace(id, "+", ""):
      tripel = tripel - 1
      retval = retval + id
  #}
  return retval
#}

def main():
  if len(sys.argv) > 1:
    key = sys.argv[1]
    print getValue(key)
  else:
    print "Usage: python secret.py <KEY>"

if __name__ == '__main__':
    main()
```

demo-qdb.py

Kommandozeilenaufruf des Servers.

```
#!/usr/bin/python

from DatabaseAPI import querydb, qclient, config, regiondb, db
import readline   # provide elaborate line editing in raw_input()
import socket
import string
import sys
import types

def main():
```

```
    if len(sys.argv) <= 1:
        sys.stderr.write("Usage: -client [ <port-key> ] | -server [ <port-key> ]\n")
        sys.stderr.write("\t%s\t%s\n" %("key","port"))
        for i in range(len(config.qdb_ports)):
            sys.stderr.write("\t%s\t%s\n" %(i, config.qdb_ports[i]))
        sys.exit(1)
    regiondb.setDefaultDbconn(db.Db())
    qdb = querydb.Query(None, 8)            # default: verbose
    try:
        if "-q" in sys.argv:                # quiet
            qdb.close()
            qdb = querydb.Query(None, 1)
            sys.argv.remove("-q")
        if len(sys.argv) > 1:
            port = sys.argv[1]
            if string.strip(port)[:2] == '-c':
                qdb.close()
                if len(sys.argv) > 2:
                    i = int(sys.argv[2])
                    port = config.qdb_ports[i]
                    qdb=qclient.Query(('',port))
                else:
                    qdb=qclient.Query()
            else:
                if string.strip(port)[:2] == '-s':
                    if len(sys.argv) > 2:
                        i = int(sys.argv[2])
                    else:
                        i = 1
                    port = config.qdb_ports[i]
                sys.stderr.write("Server auf Port "+`port`+"\n")
                qdb.close()
                server=querydb.QueryServer(int(port))
                server.serve_until_shutdown()
                server.close()
                sys.exit()
        while 1:
            sys.stdout.write("->")
            line = raw_input()
            if not line:
                break
            query, select = qdb.query(line)
            if type(qdb.unpack) == types.DictType:
                qdb.unpack[select](sys.stdout.write, query)
            else:
                qdb.unpack(sys.stdout.write, query, select)
        qdb.close()
        sys.exit()
    except socket.error, data:
        print socket.error, data
        qdb.close()
        sys.exit()

if __name__ == '__main__':
    main()
```

`tagger.py`

Wrapperklasse um den Tagger.

```
#!/usr/bin/python

# Modul tagger

from DatabaseAPI import config, cache
from Region      import absatz
import popen2
import string
import sys

tagcache = cache.DefaultCache('tagger')       # modulglobaler Cache

def tagline(line, lang):
    if tagcache.has_key((lang, line)):
        return tagcache[(lang, line)]
    r,w=popen2.popen2(getattr(config, "tagger_" + lang) +  " 2>/dev/null")
    w.write(line)
    w.write("\n<DOKENDE>\n*\n")
    w.close()
    getaggt=r.read()
    r.close()
    if getaggt[-1] == "\n":
        getaggt = getaggt[:-1]    # letztes Newline entfernen
    else:
        sys.stderr.write("Warnung: letzte Tagzeile ohne Newline\n")
    absaetze = string.split(getaggt, "<ABSATZ>\n")
    zeilenliste = string.split(absaetze[0], "\n")
    reg = absatz.AbsatzVonZeilenliste(zeilenliste[:-1])
    tagcache[(lang, line)] = reg
    return reg

```

O.3 Korpora und die Datenbank

O.3.1 `korpus2db.py`

Dieses Skript kann zum Eintragen von Korpora benutzt werden, schöner ist aber `preproc.py` (s. Kapitel K.2).

```
#!/usr/bin/python
#schreibt einen Korpus in die Datenbank
#TODO: Hacks weg - also mit mehr Tagsets umgehen koennen

import getopt
import string
import sys
import types

from DatabaseAPI import config, db, cache

class Korpus:

    def __init__(self):
        self.dbconn = db.Db()
        self.word_idx = self.dbconn.getMaxIndex("tokenid", "tokens")
        self.sent_idx = self.dbconn.getMaxIndex("satzid", "saetze")
        self.map_idx = self.dbconn.getMaxIndex("mapid", "map_s_w")
        self.segment_idx = self.dbconn.getMaxIndex("segmentnr", "map_s_w")
        self.autor_idx = self.dbconn.getMaxIndex("autorid", "autoren")
        self.lemma_idx = self.dbconn.getMaxIndex("grundformid", "grundformen")
        self.trans_table = string.maketrans("\"'`´", "####")
```

```
        self.cache = cache.DefaultCache(self)

    def txt2db(self, herkunft, tpl1, tpl2):
        segstart = self.segment_idx
        for tpl in [tpl1, tpl2]:
            file = tpl[0]
            lang = tpl[1]
            langnr = config.lang2id[lang]
            tagsetnm = tpl[2] + "_" + lang
            print "entering %s into db..." %file
            input = open(file, "r")
            lines = string.split(input.read(), "<DOKENDE>")
            segs = string.split(lines[0], "<segmentgrenze>")
            self.tag_idx = self.dbconn.getMaxIndex("tagid", tagsetnm)
            tagset = self.dbconn.executelist("select name, tagsetid from tagsets where sprachenid = %s and name
            if tagset:
                tagset = tagset[0]
            self.entersegmente(segs, self.sent_idx, herkunft, langnr, tagset)
            input.close()
            self.segment_idx = segstart
        self.dbconn.commit()

    def entersegmente(self, segmente, sentstart, herkunft, sprache, tagset):
        next = self.dbconn.incr(sentstart)
        prev = -1
        for seg_line in segmente:
            for sent in string.split(seg_line, "<SATZ>"):
                sent = string.strip(sent)
                self.entersentence(sent, prev, next, herkunft, sprache, tagset)
                prev = self.dbconn.incr(prev)
                if prev == 0:
                    prev = db.unknown2string(long(self.sent_idx) - 1)
                    next = self.dbconn.incr(next)
            self.segment_idx = self.dbconn.incr(self.segment_idx)
        self.dbconn.update("update saetze set succ = -1 where satzid = %s" % (db.unknown2string(long(self.sent_i

    def entersentence(self, sentence, prev, next, herkunft, sprache, tagset):
        #print sentence
        self.dbconn.update("insert into saetze(satzid, prev, succ, sprachenid, herkunftsid) values(%s, %s, %s, %
        koks_tag = 0
        for word in string.split(sentence, "\n"):
            word = string.strip(word)
            if word == "<KOKS>":
                if string.find(sentence, "</KOKS>") >= 0:
                    koks_tag = 1
                else:
                    break  # Schleife verlassen (sent_idx nur noch zu erhoehen)
            if koks_tag:
                if word == "</KOKS>":
                    koks_tag = 0
                continue
            self.enterword(string.translate(word, self.trans_table), tagset)
        self.sent_idx = self.dbconn.incr(self.sent_idx)

    def enterword(self, word, tagset):
        #print 'word'
        if len(string.split(word, "\t")) == 3:
            ckey = word
            if self.cache.has_key(ckey):
                wort_idx = self.cache[ckey]
            else:
```

```
                word = string.replace(word, '\\', '/')
                wort, tag, grundform = string.split(word, "\t")
                #print wort, tag, grundform
                tagset_name = tagset[0]
                tagset_idx = db.unknown2string(tagset[1])
                tag_idx = self.enterunique(tag, tagset_name, "name", "tagid", "tag_idx")
                gf_idx = self.enterunique_complex(grundform, "grundformen", "grundformid",
                    "where name = '%s' and tagid = %s and tagsetid = %s" %(grundform, tag_idx, tagset_idx),
                    "grundformid, name, reverse, tagid, tagsetid",
                    "%s, '%s', '%s', %s, %s" %(self.lemma_idx, grundform, self.p_reverse(grundform),
                                               tag_idx, tagset_idx), "lemma_idx")
                wort_idx = self.enterunique_complex(wort, "tokens", "tokenid",
                    "where name = '%s' and grundformid = %s and tagid = %s and tagsetid = %s" %(wort,
                                               gf_idx, tag_idx, tagset_idx),
                    "tokenid, name, reverse, grundformid, tagid, tagsetid",
                    "%s, '%s', '%s', %s, %s, %s" %(self.word_idx, wort, self.p_reverse(wort),
                                               gf_idx, tag_idx, tagset_idx), "word_idx")
                self.cache[ckey] = wort_idx
            self.dbconn.update("insert into map_s_w (mapid, tokenid, satzid, segmentnr) values(%s, %s, %s, %s
            self.map_idx = self.dbconn.incr(self.map_idx)

    def p_reverse(self, word):
        str = ""
        for l in word:
            str = l + str
        return str

    def enterunique(self, unique, table, column, idxcolumn, max_idx):
        return self.enterunique_complex(unique, table, idxcolumn, "where %s = '%s'" %(column, unique), "%s, %

    def enterunique_complex(self, unique, table, idxcolumn, where, columns, values, max_idx):
        idx = self.dbconn.executelist("select %s from %s %s" %(idxcolumn, table, where))
        if idx:
            idx = idx[0][0]
        else:
            self.dbconn.update("insert into %s (%s) values(%s)" %(table, columns, values))
            idx = long(getattr(self, max_idx))
            setattr(self, max_idx, db.unknown2string(idx + 1)) #unschoen, aber pass by ref ist halt nicht...
        return db.unknown2string(idx)

    def enterquelle(self, name, verlag, bemerkungen): #TODO schon drin???
        qid = self.dbconn.getMaxIndex("quellenid", "quellen")
        self.dbconn.update("insert into quellen(quellenid, name, verlag, bemerkungen) values(%s, '%s', '%s',
        return qid

    def enterherkunft(self, name, autor, jahr, quellenid):
        hid = self.dbconn.getMaxIndex("herkunftsid", "herkunft")
        aid = self.enterunique(autor, "autoren", "name", "autorid", "autor_idx")
        self.dbconn.update("insert into herkunft(herkunftsid, name, autorid, jahr, quellenid) values(%s, '%s'
        return hid

    def getinfo(self, file, infodict):
        print "Bitte machen Sie Angaben zu %s:" % file
        for inf in infodict.keys():
            if not infodict[inf]:
                print inf, '?'
                infodict[inf] = sys.stdin.readline()[:-1]
        qid = self.enterquelle(infodict['Quelle'], infodict['Verlag'], infodict['Bemerkungen'])
        hid = self.enterherkunft(infodict['Textname'], infodict['Autor'], infodict['Jahr'], qid)
        sid = self.choose("sprachen", 'name', 'sprachenid', 'Welche Sprache hat die Datei %s?'% file)
        return (hid, sid, infodict)
```

```
145
146     def choose(self, table, col, id, question):
147         langlist = self.dbconn.executelist("select %s, %s from %s" %(id, col, table))
148         nr = -1
149         while nr < 0 or not nr in map(lambda x: x[0], langlist):
150             print question
151             for row in langlist:
152                 print row[0], row[1]
153             nr = sys.stdin.readline()[:-1]
154             try:
155                 nr = int(nr)
156             except:
157                 nr = -1
158         return nr
159
160     def close(self):
161         self.dbconn.close()
162
163
164 def main():
165     usage = """%s [-e <korpus1> <korpus2>] [-h]
166            -e: enter aligned korpora
167            -h: print this help
168            the options cannot be combined!!!
169            """ %sys.argv[0]
170     try:
171         (options, args) = getopt.getopt(sys.argv[1:], "eh")
172     except:
173         print usage
174         sys.exit()
175
176     if len(args) < 1 or len(options) > 1:
177         print usage
178         sys.exit()
179
180     korpus = Korpus()
181     if options[0][0] == '-e':
182         infodict = {'Textname': '', 'Autor': '', 'Jahr': '', 'Quelle': '', 'Verlag': '', 'Bemerkungen': ''}
183         (hid, sid1, infodict) = korpus.getinfo(args[0], infodict)
184         sid2 = 3 - sid1 #Hack! Sprachen entweder 1 oder 2...
185         print "entering texts into db"
186         korpus.txt2db(hid, (args[0], sid1), (args[1], sid2))
187     else:
188         print usage
189     korpus.close()
190
191 if __name__ == '__main__':
192     main()
193
194
195
196
```

O.3.2 `delkorpus.py`

Mit diesem Skript kann man die Korpora auflisten, bzw. einzelne löschen.

```
1 #!/usr/bin/python
2
3 import sys
4 from DatabaseAPI import db
```

```
5
6   def listkorpus(dbc):
7       korpora = dbc.executelist("select h.herkunftsid, h.name, q.name, q.bemerkungen from herkunft h, quellen q
8       print "ID\tPath\tName\tNotes\n"
9       print 72 * "-"
10      for korpus in korpora:
11          print "%s\t%s\t%s\t%s\n" % (db.unknown2string(korpus[0]), korpus[1], korpus[2], korpus[3])
12      print 72 * "-"
13
14  def sel(dbc, string):
15      list = dbc.executelist(string)
16  #    print string, list
17      if not list or not list[0] or not list[0][0]:
18          return -1
19      return int(db.unknown2string(list[0][0]))
20
21  def delkorpus(dbc, id):
22      answer = raw_input("really delete korpus " + id + "??? (y/n) ")
23      if not answer in ["y", "Y"]:
24          return
25      qid = dbc.executelist("select quellenid from herkunft where herkunftsid = %s" %id)
26      if not qid:
27          print "korpus doesn't exist!!!"
28          return
29      minsatz = sel(dbc, "select min(satzid) from saetze where herkunftsid = %s" %id)
30      print "minimal sentence id %s" %minsatz
31      maxsatz = sel(dbc, "select max(satzid) from saetze where herkunftsid = %s" %id)
32      print "maximal sentence id %s" %maxsatz
33      minseg = sel(dbc, "select min(mapid) from map_s_w where satzid >= %s and satzid < %s" %(minsatz, maxsatz)
34      print "minimal segment id %s" %minseg
35      maxseg = sel(dbc, "select max(mapid) from map_s_w where satzid >= %s and satzid < %s" %(minsatz, maxsatz)
36      print "maximal segment id %s" %maxseg
37      if maxsatz == -1 or minsatz == -1 or maxseg == -1 or minseg == -1:
38          print "problem getting minimum / maximum values for deletion...quitting!"
39          return
40      dbc.update("delete from saetze where satzid between %s and %s" %(minsatz, maxsatz))
41      dbc.update("delete from map_s_w where mapid between %s and %s" %(minseg, maxseg))
42      dbc.update("delete from herkunft where herkunftsid = %s" %id)
43      if len(qid) == 1:
44          dbc.update("delete from quellen where quellenid = %s" %db.unknown2string(qid[0][0]))
45
46  def main():
47      if len(sys.argv) < 2:
48          print "Listing korpora..."
49          listkorpus(db.Db())
50          print sys.argv[0], "<korpus_id> to delete the wanted corpus"
51          sys.exit(0)
52      else:
53          delkorpus(db.Db(), sys.argv[1])
54          sys.exit(0)
55
56  if __name__ == '__main__':
57      main()
```

Anhang P

Demo-Code

P.1 Code für die Demo-Application

P.1.1 Java-Applet KoksAppe.java

```
package koks;

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.HttpURLConnection ;
import java.net.URL ;
import java.io.IOException;
import java.net.*;

/**
 * Applet
 * <P>
 * @author Norman Kummer
 * @version 0.2 - 15.06.2001
 *
 * @author Arno Erpenbeck
 * @version 0.3 - 21.06.2001
 *
 * @author Norman Kummer
 * @version 0.4 23.08.2001 (english)
 *
 * @author Arno Erpenbeck
 * @version 0.4.1 - 23.08.2001
 *
 * @author Arno Erpenbeck
 * @version 0.4.2 - 08.10.2001
 */
public class KoksAppe extends Applet {

//Klasse, welche die Oberfläche abbildet

  private static final String VERSION = "0.4.2";
  //private static final String BASE_URL =
"http://nulix.cl-ki.uni-osnabrueck.de/~koks/cgi-bin/hit1e.rb";
  private String baseURL;
  private static final int MAX = 10;

  boolean isStandalone = false;
  TextArea textArea1 = new TextArea(5, 100);
  TextArea textArea2 = new TextArea(2, 100);
  TextField textArea3 = new TextField(Integer.toString(MAX), 2);
```

```
43    Button button1 = new Button();
44    Button button2 = new Button();
45    BorderLayout borderLayout1 = new BorderLayout();
46  //Variable zum Sprachenumschalten (dtl./engl.)
47    Choice cSprache = new Choice();
48  //protokollieren ja/nein
49    Checkbox checkBox = new Checkbox("pro", false);
50    /**
51     * getParameter
52     * @param key
53     * @param def
54     * @return java.lang.String
55     */
56    public String getParameter(String key, String def) {
57      if (isStandalone) {
58        return System.getProperty(key, def);
59      }
60      if (getParameter(key) != null) {
61        return getParameter(key);
62      }
63      return def;
64    }
65
66    /**
67     * Constructs a new instance.
68     */
69    public KoksAppe() { }
70
71    /**
72     * init
73     */
74    public void init() {
75      try  {
76        jbInit();
77      }
78      catch (Exception e) {
79        e.printStackTrace();
80      }
81    }
82
83    /**
84     * Initializes the state of this instance.
85     */
86    private void jbInit() throws Exception {
87      // URL fuer Server
88  //URL zum 1. Ruby-Script
89      baseURL = getParameter("baseURL",
90  "http://nulix.cl-ki.uni-osnabrueck.de/~koks/cgi-bin/hit1e.rb");
91      // Layout und Farbe
92      this.setLayout(borderLayout1);
93      this.setBackground(Color.lightGray);
94      // set Font
95          int size = Integer.parseInt(getParameter("fontsize", "24"));
96          Font font = new Font("SansSerif", Font.PLAIN, size);
97      this.setFont(font);
98      // Start Button
99      button1.setLabel("Start");
100     //button1.setBackground(Color.green);
101     button1.addActionListener(new java.awt.event.ActionListener() {
102       public void actionPerformed(ActionEvent e) {
103         button1_actionPerformed(e);
```

```
104       }
105     });
106     // Reset Button
107     button2.setLabel("Reset");
108     //button2.setBackground(Color.red);
109     button2.addActionListener(new java.awt.event.ActionListener() {
110       public void actionPerformed(ActionEvent e) {
111         button2_actionPerformed(e);
112       }
113     });
114     // Eingabefeld für Sätze
115     //textArea1.setText("Wir werden jemanden in Kenntnis setzen.");
116     textArea1.setBackground(Color.white);
117     // Ausgabefeld für markierten Satz
118     //textArea2.setBackground(Color.white);
119     //textArea2.setEditable(false);
120     // DropDown für Sprachauswahl
121     cSprache.add("G");
122     cSprache.add("E");
123     // unteres Panel enthält Button und DropDown
124     Panel unten = new Panel(new GridLayout(1, 2));
125     Panel links = new Panel(new GridLayout(1, 6, 8, 8));
126     links.add(new Label(VERSION));
127     links.add(new Label("lang: "));
128     links.add(cSprache);
129     links.add(new Label("hits: "));
130     links.add(textArea3);
131     links.add(checkBox);
132     Panel rechts = new Panel(new GridLayout(1, 2));
133     rechts.add(button2);
134     rechts.add(button1);
135     unten.add(links);
136     unten.add(rechts);
137     // alles einbauen
138     this.add(unten,BorderLayout.SOUTH);
139     this.add(textArea1, BorderLayout.CENTER);
140     //this.add(textArea2, BorderLayout.CENTER);
141   }
142
143   /**
144    * getAppletInfo
145    * @return java.lang.String
146    */
147   public String getAppletInfo() {
148     return "Applet Information";
149   }
150
151   /**
152    * getParameterInfo
153    * @return java.lang.String[][]
154    */
155   public String[][] getParameterInfo() {
156     return null;
157   }
158
159   void button2_actionPerformed(ActionEvent e) {
160     //textArea1.setText("Wir werden jemanden in Kenntnis setzen");
161     textArea1.setText("");
162     //textArea2.setText("");
163     textArea3.setText(Integer.toString(MAX));
164   }
```

```
165
166   void button1_actionPerformed(ActionEvent e) {
167     // Satz herausfiltern zwischen .!?"
168     int anfang, ende, max = MAX;
169     String text = textArea1.getText();
170     String selected = textArea1.getSelectedText();
171     String wort = new String(selected.trim());
172     String phrase, sprache;
173     int pos = textArea1.getCaretPosition();
174     int wortAnfang = textArea1.getSelectionStart();
175     int wortEnde = textArea1.getSelectionEnd();
176     String vorCursor = new String(text.substring(0, pos));
177     String nachCursor = new String(text.substring(pos, text.length()));
178
179     // Anfang und Ende vom Satz suchen
180     anfang = vorCursor.lastIndexOf(".") + 1;
181
182     if (anfang < vorCursor.lastIndexOf("!"))
183       anfang = vorCursor.lastIndexOf("!") + 1;
184     if (anfang < vorCursor.lastIndexOf("?"))
185       anfang = vorCursor.lastIndexOf("?") + 1;
186     if (anfang < vorCursor.lastIndexOf("’"))
187       anfang = vorCursor.lastIndexOf("’") + 1;
188
189     ende = vorCursor.length() + nachCursor.indexOf(".") + 1;
190
191     if ((ende > nachCursor.indexOf("!")) &&
192         (nachCursor.indexOf("!") != -1))
193       ende = nachCursor.indexOf("!");
194     if ((ende > nachCursor.indexOf("?")) &&
195         (nachCursor.indexOf("?") != -1))
196       ende = nachCursor.indexOf("?");
197     if ((ende > nachCursor.indexOf("’")) &&
198         (nachCursor.indexOf("’") != -1))
199       ende = nachCursor.indexOf("’");
200
201     //int wortAnfang = textArea1.getSelectionStart() - anfang;
202     //int wortEnde = textArea1.getSelectionEnd() - ende;
203 //Phrase extrahieren
204     phrase  =  cleanString(new String(text.substring(anfang, wortAnfang) +
205       "<i> " + text.substring(wortAnfang, wortEnde) + " </i> " +
206       text.substring(wortEnde, ende)));
207
208     //textArea2.setText("Anfang: " + anfang + " Ende: " + ende + "\n" +
209     //  "WortAnfang: " + wortAnfang + " WortEnde: " + wortEnde + "\n" +
210     //  "SelStart: " + textArea1.getSelectionStart() + " SelEnd: " +
211     //  textArea1.getSelectionEnd() + "\n");
212 //Satz markieren
213     //textArea1.select(anfang, vorCursor.length() + ende);
214     textArea1.select(anfang, ende);
215     //textArea2.setText("Wort: " + wort + "\nPhrase: " + phrase);
216
217     // Auswahl von Sprache auswerten
218     sprache = new String(cSprache.getSelectedItem() == "G" ? "de" : "en");
219     if (!textArea3.getText().equals(""))
220
221     // max. Trefferanzahl auswerten
222     try {
223       max = Integer.parseInt(textArea3.getText());
224     } catch (NumberFormatException nfe) {
225       max = MAX;
```

```
    }
        max = (max <= 0) ? MAX : max;

    // URL öffnen und in neuem Frame anzeigen
    try {
      getAppletContext().showDocument(createURL(phrase, sprache, max),
"ausgabe1");
    }
    catch (Exception ex) {
      //textArea2.setText("Exception: " + e.toString());
      ex.printStackTrace();
    }
  }

  private URL createURL(String phrase, String sprache, int max) throws Exception
{
//erstellt eine neue URL mit den Parametern
    String verbose = checkBox.getState() ? "1" : "0";
    URL url = new URL(baseURL + "?port=29295&phrases="+
URLEncoder.encode(phrase) + "&max=" + max +
"&t=t:segment&s=s:text&applet=true&lang=" + sprache + "&verbose=" + verbose);
    return url;
  }

  private String cleanString(String phrase) {
    String result = phrase.replace('\t', ' ').replace('\n', ' ').trim();
    return result;
  }

}
```

P.1.2 hit1e.rb

Anfrage vom Applet bearbeiten (s. 10.3)

```
#!/usr/local/bin/ruby

# $Header: /opt/cvs/doc/phase_1/demo/hit1e.rb,v 1.2 2001/10/23 17:27:57 koks-cvs Exp $

# hit1.rb - Process queries from web frontend and ask
#           demo-qdb for results
#           Adapted version for HIT
#           Adapted version for EuroCALL
#
# KOKS project
# Arno Erpenbeck, 06.08.2001
# english by Norman Kummer, 21.08.2001

require "cgi"
require "socket"
require "base64"
require "md5"
require "secret"

$dbURL  = "nulix.cl-ki.uni-osnabrueck.de"
$cgiURL = "http://nulix.cl-ki.uni-osnabrueck.de/~koks/cgi-bin/hit2e.rb"

class Ergebnis
  include Comparable

  attr_reader :nr, :id, :type, :lemmata
```

```
27    attr_accessor :bid
28
29    def initialize(nr, id, type="unknown")
30      @nr = nr
31      @id = id
32      @type = type
33      @tokens = []
34      @lemmata = []
35    end
36
37    def size
38      @lemmata.size
39    end
40
41    def <=>(anOther)
42      self.size <=> anOther.size
43    end
44
45    def add_lemma(lem)
46      @lemmata << lem
47    end
48
49    def add_token(tok)
50      @tokens << tok
51    end
52
53    def to_s
54      str = "<tr><td valign=\"top\">\n"
55      if @type == "wb"
56        str << "<input type=\"hidden\" name=\"lemmata_#{@nr}\" value=\"" +
57  @lemmata.join(" ") + "\">\n"
58      elsif @type == "erkannt"
59        str << "<input type=\"hidden\" name=\"bid_#{@nr}\" value=\"#{@id}\">\n"
60      end
61      str << "<input type=\"hidden\" name=\"type_#{@nr}\" value=\"#{@type}\">\n"
62      str << "<input type=\"radio\" name=\"select\" value=\"#{@nr}\"></td>\n"
63      str << "<td><tt>"
64      @tokens.each do |tok|
65        str << "#{CGI.escapeHTML(tok)} "
66      end
67      str << "</tt></td></tr>\n"
68      str
69    end
70
71  end
72
73  def askServer(port, cmd=nul)
74    begin
75      # open socket to specified server
76      socket = TCPSocket.open($dbURL, port)
77      # wait for challenge
78      question = socket.gets
79      what = question.split("'")[1]
80      # compute secret and send to server
81      socket.print what.computeSecret, "\n"
82      socket.flush
83      resp = socket.gets
84      if resp.strip! != "OK"
85        raise "Authentication not accepted"
86      end
87      # send command
```

```
88      socket.print cmd
89      socket.flush
90      puts "<form name=\"myform\" action=\"#{$cgiURL}\" target=\"ausgabe2\"
91  method=\"post\">"
92      puts "<input type=\"hidden\" name=\"port\" value=\"#{port}\">"
93      puts "<input type=\"hidden\" name=\"t\" value=\"t:segment\">"
94      puts "<input type=\"hidden\" name=\"s\" value=\"s:text\">"
95      # read until finished
96      count = 1
97      id = ""
98      type = ""
99      modus = :anfang
100     lemma = []
101     erkannt = []
102     wb = []
103     ergebnis = nil
104     while resp = socket.gets.strip
105       if resp == "<DONE>" || resp == "<ERROR>"
106         break
107       elsif resp == "<SATZ>" || resp == "<segmentgrenze>" ||
108             resp == "<unbestimmter Regionentyp>"
109         if ergebnis.type == "erkannt"
110           erkannt << ergebnis
111         elsif ergebnis.type == "wb"
112           wb << ergebnis
113         else
114         end
115         count += 1
116         modus = :warte
117         lemma = []
118       elsif resp =~ /(<id:)(\d+) (type:)(\w+)/
119         id = $2
120         type = $4
121         ergebnis = Ergebnis.new(count, id, type)
122       elsif resp =~ /(<id:)(\d+)/
123         id = $2
124         ergebnis = Ergebnis.new(count, id, type)
125       else
126         tokens = resp.split('\t')
127         ergebnis.add_token(tokens[0])
128         ergebnis.add_lemma(tokens[2])
129       end
130     end
131
132     puts "<h4>according to examples</h4>"
133     puts "<table border=\"0\">"
134     erkannt.sort.each do |e| puts e end
135     puts "</table>"
136
137     puts "<h4>according to the dictionary</h4>"
138     puts "<table border=\"0\">"
139     wb.sort.each do |w| puts w end
140     puts "</table>"
141
142     puts "<br>number of hits: <input type=\"text\" size=\"10\" name=\"max\"
143 value=\"10\">"
144     puts " <input type=\"submit\" value=\"look at it\">\n</form>"
145     # signal end of data
146     socket.print "quit\n"
147     socket.flush
148   rescue
```

```
149     puts "<font color=\"red\">Error connecting to server: #{$!}</font>"
150     puts "</body></html>"
151     return
152   ensure
153     socket.close if socket
154   end
155 end
156 
157 # print out some info
158 cgi = CGI.new
159 print "Content-type: text/html\r\n\r\n"
160 puts "<html><head><meta http-equiv=\"expires\" content=\"0\"></head>"
161 puts "<body bgcolor=\"white\">"
162 
163 # process form values
164 verbose = cgi.params["verbose"].to_s == "1"
165 port = cgi.params["port"].to_s
166 if cgi.params["wort"].to_s.length == 0 && cgi.params["phrases"].to_s.length == 0
167   puts "<font color=\"red\">Error: look up for the empty word!</font>"
168   puts "</body></html>"
169   exit
170 end
171 
172 cmd = ""
173 cmd << cgi.params["t"].to_s + ";"
174 cmd << cgi.params["s"].to_s + ";w:"
175 ["lang", "phrases", "id", "lemmata", "match", "name", "wort", "max",
176 "satzid"].each { |c|
177   if cgi.params[c].to_s.length > 0
178     str = cgi.params[c].to_s
179     str = str.delete(",") if c == "phrases"
180     while str.gsub!(/  /, " ")
181     end
182     cmd << c + "=" + str + ","
183   end
184 }
185 cmd.chop! if cmd[-1,1] == ","
186 
187 if verbose
188   puts "Port: <tt>", port, "</tt><br>"
189   puts "Command: <tt>", cmd, "</tt><br>"
190 end
191 puts "<h3>phrases found:</h3>"
192 STDOUT.flush
193 
194 askServer(port.to_i, cmd)
195 
196 # close document
197 puts "</body></html>"
198 
199 
```

P.1.3 hit2e.rb

Anfrage vom Applet bearbeiten (s. 10.3)

```
1 #!/usr/local/bin/ruby
2 
3 # $Header: /opt/cvs/align/frontend/hit2.rb,v 1.1 2001/06/21 13:28:48 koks-cvs
4 Exp $
5 
```

```
# hit2.rb - Process queries from web frontend and ask
#           demo-qdb for results
#           Adapted version for HIT
#           Adapted version for EuroCALL
#
# KOKS project
# Arno Erpenbeck, 06.08.2001
# english by Norman Kummer, 21.08.2001

require ''cgi''
require ''socket''
require ''base64''
require ''md5''
require ''secret''

$dbURL  = ''nulix.cl-ki.uni-osnabrueck.de''

def askServer(port, cmd=nul)
  begin
    # open socket to specified server
    socket = TCPSocket.open($dbURL, port)
    # wait for challenge
    question = socket.gets
    what = question.split(''''')[1]
    # compute secret and send to server
    socket.print what.computeSecret, ''\n''
    socket.flush
    resp = socket.gets
    if resp.strip! != ''OK''
      raise ''Authentication not accepted''
    end
    # send command
    socket.print cmd
    socket.flush
    # read until finished
    count = 1
    id = ''''
    lemma = []
    deutsch = true
    while resp = socket.gets.strip
      if resp == ''<DONE>'' || resp == ''<ERROR>''
        break
      elsif resp == ''<SATZ>'' || resp == ''<segmentgrenze>'' ||
            resp == ''<unbestimmter Regionentyp>''
        count += 1
        lemma = []
      elsif resp =~ /(<id:)(\d+)(>)/
        id = $2
        puts deutsch ? ''<br><br>german: '' : ''<br>english: ''
        deutsch = !deutsch
      else
        tokens = resp.split('\t')
        lemma << tokens[2]
        puts ''<tt>#{CGI.escapeHTML(tokens[0])} </tt>''
      end
    end
    # signal end of data
    socket.print ''quit\n''
    socket.flush
  rescue
    puts ''<font color=\''red\''>Error connecting to server: #{$!}</font>''
```

```
    puts ''</body></html>''
    return
  ensure
    socket.close if socket
  end
end

# print out some info
cgi = CGI.new
print ''Content-type: text/html\r\n\r\n''
puts ''<html><head><meta http-equiv=\''expires\'' content=\''0\''></head>''
puts ''<body bgcolor=\''white\''>''

# process form values
port = cgi.params[''port''].to_s
select = cgi.params[''select''].to_s.to_i
bid = cgi.params[''bid_#{select}''].to_s
lemmata = cgi.params[''lemmata_#{select}''].to_s
if bid.length == 0 && lemmata.length == 0
  puts ''<font color=\''red\''>Error: look up without BID or lexical item!</font>''
  puts ''</body></html>''
  exit
end

cmd = ''''
cmd << cgi.params[''t''].to_s + '';''
cmd << cgi.params[''s''].to_s + '';w:''
[''lang'', ''phrases'', ''id'', ''lemmata'', ''match'', ''name'', ''wort'', ''max'',
''satzid''].each { |c|
  if cgi.params[c].to_s.length > 0
    str = cgi.params[c].to_s
    while str.gsub!(/  /, '' '')
    end
    cmd << c + ''='' + str + '',''
  end
}

if select > 0
  if lemmata.length > 0
    #cmd << ''match_l='' + lemmata
    cmd << ''lemmata='' + lemmata
  else
    cmd << ''bid='' + bid if select > 0
  end
end
cmd.chop! if cmd[-1,1] == '','' 

#puts ''Command: <tt>'', cmd, ''</tt><br>''
puts ''<h3>translations found:</h3>''
STDOUT.flush

askServer(port.to_i, cmd)

# close document
puts ''</body></html>''
```