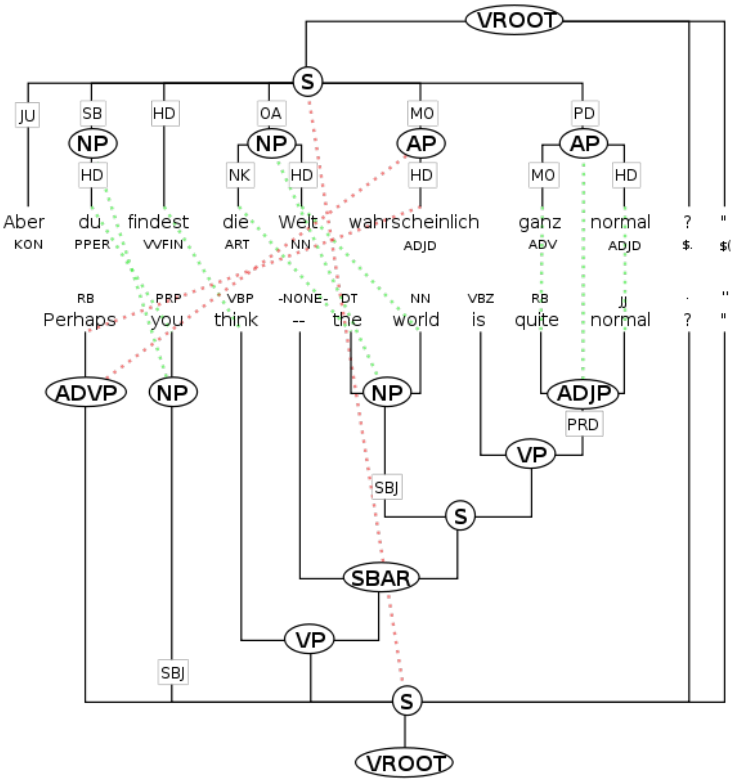
Universität Zürich
Computerlinguistik

Automatisches Alignieren in parallelen Baumbanken



Lizenziatsarbeit der Philosophischen Fakultät der Universität Zürich
Referent: Prof. Dr. M. Volk

Verfasserin:
Sandra Roth
Blumenfeldstrasse 5
8046 Zürich

9. November 2009

Zusammenfassung

Parallele Baumbanken sind eine immer wichtiger werdende Ressource im Bereich der maschinellen Übersetzung. Ihre Erstellung ist aber ein langwieriger Prozess, und um eine gute Qualität gewährleisten zu können, müssen viele Schritte manuell gemacht oder zumindest manuell überprüft werden. Mit dem TreeAligner wurde ein Werkzeug geschaffen, mit dem sich parallele Baumbanken manuell alignieren und durchsuchen lassen. Um das Alignieren für den menschlichen Annotator zu beschleunigen und konstanter zu halten, wurde der TreeAligner im Rahmen dieser Arbeit mit automatischen Vorschlägen für Alignierungen erweitert.

Abstract

Parallel treebanks are an increasingly important ressource in the area of machine translation. Their creation though is a tedious process. In order to achieve good quality, many steps must be done, or at least verified, manually. TreeAligner is a tool to align, browse and search parallel treebanks manually. To speed up the task for the human annotator and improve consistency, this project extended TreeAligner with automatic proposals for alignments.

Danksagung

Ich bedanke mich ganz herzlich bei Prof. Dr. Martin Volk für die gute Betreuung im Verlaufe dieser Arbeit und für den mitreissenden Enthusiasmus in Bezug auf Baumbanken, maschinelle Übersetzung und den TreeAligner.
Ein weiterer Dank geht an Torsten Marek, den Hauptentwickler des TreeAligners, der mir eine Einführung in die Programmstruktur des TreeAligners gegeben und auch während der Programmierphase immer geduldig meine Fragen beantwortet hat.
Zu guter Letzt danke ich auch Anne Göhring, Irène Roth und Oli Schacher für das Korrekturlesen und die moralische Unterstützung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASCII	American Standard Code for Information Interchange
CAT	Computer-Aided Translation
ID	Identifikator
LDC	Linguistic Data Consortium
LFG	Lexikalisch-funktionale Grammatik (Lexical Functional Grammar)
NP	Nominalphrase
NT	Nonterminal
POS	Part of Speech
PP	Präpositionalphrase
S	Satz
SMT	Statistical Machine Translation
STTS	Stuttgart-Tübingen-Tagset
SQL	Structured Query Language
T	Terminal
VP	Verbalphrase

1 Einleitung

1.1 Ausgangslage

Aber du findest die Welt wahrscheinlich ganz normal? - Perhaps you think the world is quite normal? Dies sind die deutsche und englische Entsprechung eines Satzes aus Jostein Gaarders *Sofies Welt* (norwegischer Originaltitel: *Sofies verden*). Auf der Titelseite sieht man die beiden Sätze und die dazugehörigen Phrasenstrukturbäume sowie farbig gestrichelte Linien zwischen den Wörtern und Phrasen. Dies sind die sogenannten Alignierungen. Sie wurden in diesem Fall von einem menschlichen Annotator erstellt und sollen sinngemässe Entsprechungen in den beiden Sätzen kennzeichnen. Die rote Linie vom deutschen zum englischen Satzknoten lässt erkennen, dass der Annotator die Sätze nicht als genaue Übersetzung voneinander ansieht, sondern nur als eine ungefähre, während die grünen Linien die genauen Übersetzungen darstellen. Dem stimmen wir intuitiv wohl auch zu: Gewisse Wörter und Phrasen der beiden Sätze entprechen sich ziemlich genau, für andere gibt es jedoch keine genauen oder gar keine Entsprechungen. In der maschinellen Übersetzung wurde in den letzten Jahren vermehrt versucht, syntaktische Informationen, wie sie in den beiden Syntaxbäumen des Titelbilds gegeben sind, miteinzubeziehen und so zu besseren Resultaten zu gelangen (vgl. z.B. die datenorientierte Übersetzung in Hearne und Way (2003)). Dafür braucht es aber Ressourcen, parallele, geparste Korpora, welche über Alignierungen miteinander in Beziehung gesetzt werden. Dies sind die sogenannten parallelen Baumbanken. Das Erstellen einer solchen parallelen Baumbank ist jedoch ein grosser Aufwand: Parallele Texte müssen für die gewünschten Sprachpaare gefunden, getaggt, geparst und aligniert werden, je nachdem auch manuell. Um das Alignieren und dessen Automatisierung soll es in dieser Arbeit gehen.

1.2 Fragestellung, Ziel und Aufbau der Arbeit

Alignierungen für parallele Baumbanken zu erstellen, ist auch für einen geübten Annotator noch zeitaufwändig. Der TreeAligner ist ein Werkzeug, um parallele Baum-

banken (manuell) zu alignieren und Anfragen an die parallele Baumbank zu stellen. Um die Arbeit des manuellen Alignierens zu beschleunigen und auch um die Konsistenz der Alignierungen zu erhöhen, ist es das Ziel dieser Arbeit, den TreeAligner mit automatisch generierten Vorschlägen für Alignierungen zu erweitern. Dafür sollen zunächst keine externen Hilfsmittel (wie beispielsweise ein zweisprachiges Wörterbuch oder Morphologieanalyse) eingebunden werden, sondern es sollen nur die Daten der bereits manuell erstellten Alignierungen verwendet werden.
Die Fragestellung, die in dieser Arbeit beantwortet werden soll, ist, ob es möglich ist, nur mit den Informationen aus den Baumbanken und den bereits vorhandenen Alignierungsdaten gute Alignierungsvorschläge zu generieren.
Der Schwerpunkt liegt vor allem darauf, eine hohe Präzision (engl. '*precision*') der Vorschläge zu erreichen; der Annotator soll in seiner Arbeit unterstützt werden und möglichst wenig Zeit damit verbringen, Fehler zu entfernen. Die Ausbeute (engl. '*recall*') ist weniger wichtig; je mehr Daten vorhanden sind, desto grösser sollte die Ausbeute sein.
Im folgenden Kapitel 2 werden zunächst einige wichtige Baumbanken näher vorgestellt. Das Kapitel 3 gibt einen Überblick über automatische Alignierung, unterteilt nach Satz-, Wort- und Phrasenalignierung. In Kapitel 4 stelle ich die Funktionen des TreeAligners vor. Die Implementierung der automatischen Vorschläge und deren Evaluation wird in Kapitel 5 behandelt. Das 6. Kapitel fasst die wichtigsten Erkenntnisse zusammen und gibt einen Ausblick auf mögliche Weiterentwicklungen und Verbesserungen des Systems.

2 Baumbanken

2.1 Was sind Baumbanken?

Baumbanken (‘*treebanks*’) sind Sammlungen von syntaktisch annotierten Sätzen, die üblicherweise als Baumstrukturen dargestellt werden. Elektronisch verfügbare Korpora, welche linguistische Informationen enthalten, sind sowohl für die Korpuslinguistik als auch für die Computerlinguistik sehr wichtig geworden. Die ersten elektronisch verfügbaren Korpora wurden gegen Ende der 60er Jahre erstellt, wovon das erste grosse, allgemeine Korpus das Brown-Korpus (*The Brown Corpus of Standard American English*) ist[1]. Dieses Korpus umfasst Beispiele aus 15 verschiedenen Textkategorien und besteht aus etwa einer Million Wörtern des amerikanischen Englisch. Zu Beginn beinhaltete das Brown-Korpus nur die Wörter mit dazugehörigen Identifikatoren, später wurden Part-of-Speech-Tags hinzugefügt. Ohne syntaktische Information gibt ein Korpus nur Aufschluss über jene linguistischen Fragestellungen, die über die Suche einzelner Wörter zugänglich sind. Untersuchungen zu Passivstrukturen ohne Agens oder Subjektinversion sind nicht möglich (vgl. Abeillé 2003: XIII). Die ersten geparsten Korpora – also Baumbanken – fürs Englische entstanden erst rund 15 Jahre nach dem Brown-Korpus: die Lancaster-Treebank, welche zwischen 1983 und 1986 erstellt wurde, und die Penn-Treebank (vgl. Skut et al. 1998: 1).
In den folgenden Unterkapiteln werden die Penn-Treebank, wohl die wichtigste Baumbank für Englisch, das Negra-Korpus, die erste deutsche Baumbank, und die Tiger-Baumbank näher betrachtet. Im Kapitel 2.1.4 werden zwei Abfragewerkzeuge für Baumbanken vorgestellt.

2.1.1 Die Penn-Treebank

Die Penn-Treebank wurde in drei Phasen zwischen 1989 und 2000 an der University of Pennsylvania erstellt und “gilt als Vorreiter für syntaktisch annotierte Korpora”(Lezius 2001: 417-418). Sie umfasst über 4 Millionen annotierte Wortformen aus

[1]Das Handbuch zum Brown-Korpus: `http://icame.uib.no/brown/bcm.html`

verschiedensten Textsorten, von IBM-Computerhandbüchern über Artikel des Wall Street Journals bis hin zu transkribierten Telefongesprächen. Das verwendete POS-Tagset basiert auf demjenigen des Brownkorpus, wurde jedoch reduziert, um lexikalische und syntaktische Redundanzen zu vermeiden (Tabelle mit Penn-Treebank POS-Tagset siehe Anhang A.1, S.78). Sowohl das Tagging, als auch das Parsing wurden in einem ersten Schritt automatisch durchgeführt. Die Ausgabe wurde danach manuell korrigiert. Für das Parsing des Korpus wurde Fidditch verwendet. Dies ist ein deterministischer Parser, welcher von Donald Hindle entwickelt wurde. Die Ausgabe des Parsers ist durch eine Klammerstruktur dargestellt, der im Nachhinein noch POS-Tags hinzugefügt wurden (syntaktisches Tagset der Penn-Treebank siehe Anhang). In einer zweiten Phase, dem Treebank II Style, wurde das Format so erweitert, dass Prädikat-Argument-Strukturen dargestellt werden können. Dazu wurde ein kleines Set funktionaler Tags für Textkategorien, grammatische Funktionen und semantische Rollen definiert. Bis zu vier dieser funktionalen Tags können einer Konstituente hinzugefügt werden, Indices inbegriffen. Die Indices werden verwendet, um auf Koreferenzen, Spuren (*T*) und Nullkonstituenten zu verweisen (vgl. Taylor et al. 2003).

(1) Analysts said Mr. Stronach wants to resume a more influential role in running the company.

Der Beispielsatz (1) aus der Penn-Treebank sieht im Klammerformat folgendermassen aus:

```
( (S
    (NP-SBJ (NNS Analysts) )
    (VP (VBD said)
      (SBAR (-NONE- 0)
        (S
          (NP-SBJ-1 (NNP Mr.) (NNP Stronach) )
          (VP (VBZ wants)
            (S
              (NP-SBJ (-NONE- *-1) )
              (VP (TO to)
                (VP (VB resume)
                  (NP
                    (NP (DT a)
                      (ADJP (RBR more) (JJ influential) )
                      (NN role) )
                    (PP-LOC (IN in)
                      (S-NOM
                        (NP-SBJ (-NONE- *) )
                        (VP (VBG running)
                          (NP (DT the) (NN company) ))))))))))
    (. .) ))
```

Damit die Struktur dieses Penn-Treebank-Beispiels etwas klarer wird, ist sie in Abbildung 2.1 noch als graphischer Baum dargestellt.

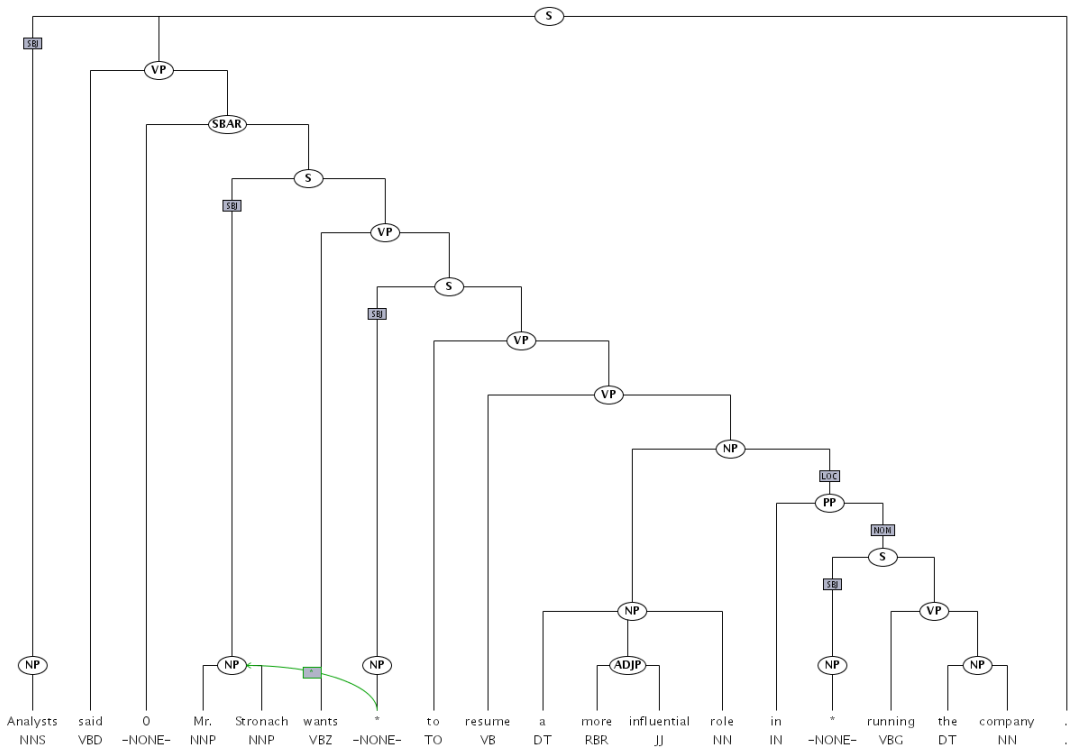


Abbildung 2.1: Beispielsatz aus der Penn-Treebank als Tiger-Graph

Die Funktionen sind in der Baumstruktur als Kantenlabels realisiert, z.B. ist die NP *Analysts* als Subjekt des Hauptsatzes S ersichtlich. Die Verbindung von der Nullkonstituente zum Subjekt, die im Klammerformat mit dem Index 1 dargestellt ist, ist im Baum als grüne Verbindung von der Nullkonstituente $*$ zum Subjekt *Mr. Stronach* realisiert.
Die Penn-Treebank ist über das Linguistic Data Consortium (LDC)[2] erhältlich, allerdings nicht kostenlos.

2.1.2 Das Negra-Korpus

Das Negra-Korpus[3] besteht aus 355'096 Tokens (20'602 Sätzen) von deutschen Zeitungstexten aus der *Frankfurter Rundschau*. Es basiert auf ca. 60'000 Tokens, die am Institut für maschinelle Sprachverarbeitung, Stuttgart, mit Parts-of-Speech annotiert wurden. Dieses Korpus wurde erweitert, ebenfalls mit Parts-of-Speech ver-

[2] `http://www.ldc.upenn.edu/`

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

sehen und vollständig mit syntaktischen Strukturen annotiert. Das Negra-Korpus wird im zeilenbasierten Export-Format dargestellt, welches sich gut zur Speicherung in Datenbanken eignet. Im Gegensatz zum Penn-Treebank-Format sind im Negra-Korpus überschneidende Kanten erlaubt. So können lokale und nicht-lokale Abhängigkeiten dargestellt werden, ohne dass Spuren (‘*traces*’) nötig sind. Kreuzende Kanten können aber automatisch in Spuren umgewandelt werden, um das Format demjenigen der Penn-Treebank anzugleichen.

(2) Selten nehmen die grossen Plattenfirmen das finanzielle Risiko auf sich, den Arbeiten zu einer Veröffentlichung zu verhelfen.

Der Beispielsatz (2) im Negra-Format:

```
#BOS 7 5 861697809 1
Selten                  ADJD    Pos                     MO      507
nehmen                  VVFIN   3.Pl.Pres.Ind           HD      507
die                     ART     Def.*.Nom.Pl            NK      500
großen                  ADJA    Pos.*.Nom.Pl.Sw         NK      500
Plattenfirmen           NN      Fem.Nom.Pl.*            NK      500
das                     ART     Def.Neut.Akk.Sg         NK      506
finanzielle             ADJA    Pos.Neut.Akk.Sg.Sw      NK      506
Risiko                  NN      Neut.Akk.Sg.*           NK      506
auf                     APPR    --                      AC      501
sich                    PRF     3.Akk.Pl                NK      501
,                       $,      --                      --      0
den                     ART     Def.*.Dat.Pl            NK      502
Arbeiten                NN      Fem.Dat.Pl.*            NK      502
zu                      APPR    Dat                     AC      503
einer                   ART     Indef.Fem.Dat.Sg        NK      503
Veröffentlichung        NN      Fem.Dat.Sg.*            NK      503
zu                      PTKZU   --                      PM      504
verhelfen               VVINF   --                      HD      504
.                       $.      --                      --      0
#500                    NP      --                      SB      507
#501                    PP      --                      MO      507
#502                    NP      --                      DA      505
#503                    PP      --                      MO      505
#504                    VZ      --                      HD      505
#505                    VP      --                      OC      506
#506                    NP      --                      OA      507
#507                    S       --                      --      0
#EOS 7
```

Die erste Spalte enthält die Knoten, d.h. die Wörter (*Selten*) oder die Nummer der Nonterminalknoten[4] (`#500`), die zweite Spalte deren Tags (`ADJD`: Adjektiv). In der dritten Spalte wird bei Terminals die Morphologieinformation (`Pos`: Positiv)

[4]Als Nonterminals (NT) werden alle Knoten bezeichnet, die nicht auf Wortebene sind, im Gegensatz zu den Terminalknoten (T).

gespeichert. Die vierte Spalte enthält die Information über die Funktion des Knotens (`MO`: Modifier), und in der fünften Spalte wird die Nummer des Elternknotens (`507`) gespeichert. Durch diesen Verweis wird die Graphstruktur repräsentiert. Die sechste Spalte kann Informationen über sekundäre Kanten beinhalten, die letzte Spalte ist für Kommentare vorgesehen (die Spalten 6 und 7 sind im obigen Beispiel nicht vorhanden). Die oberste und unterste Zeile kennzeichnen jeweils den Beginn bzw. das Ende eines Satzes. Abbildung 2.2 zeigt den Satz (2) als graphische Baumdarstellung.

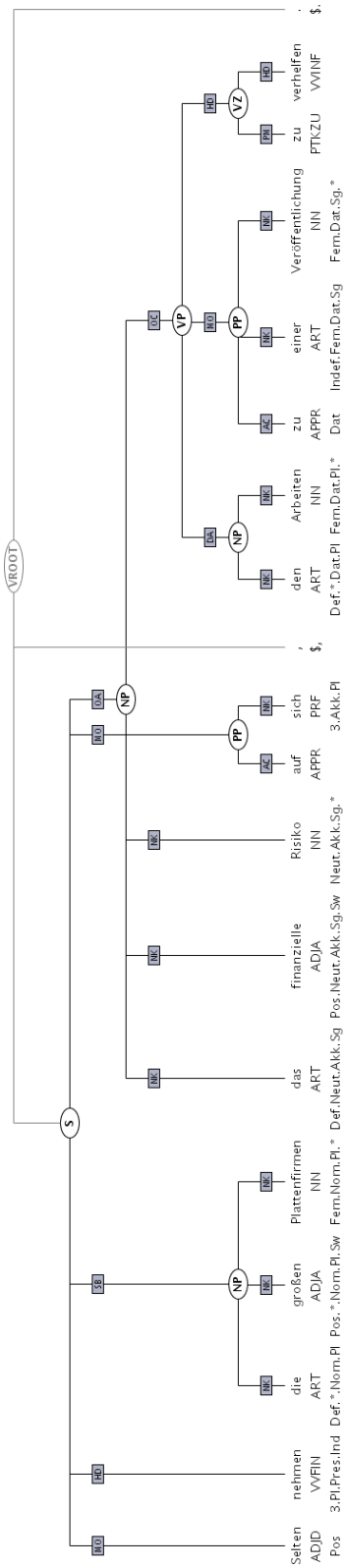


Abbildung 2.2: Beispielsatz aus dem Negra-Korpus als Tiger-Graph

2.1.3 Die Tiger-Baumbank

Neben dem Negra-Korpus ist das Tiger-Korpus[5] die bekannteste Baumbank für die deutsche Sprache. Dieses Korpus entstand zwischen 1999 und 2004 und ist ein Gemeinschaftsprojekt des Institut für Computerlinguistik und Phonetik der Universität Saarbrücken, dem Institut für maschinelle Sprachverarbeitung (IMS) der Universität Stuttgart sowie dem Institut für Germanistik der Universität Potsdam. Die Version 2.1 umfasst 50'000 Sätze, die aus ca. 900'000 Tokens bestehen, also mehr als doppelt so viele wie das Negra-Korpus. Als Basis für das Tiger-Korpus dienten ebenfalls Zeitungstexte der *Frankfurter Rundschau*.
Wie auch für das Negra-Korpus enthält das Tiger-Korpus sowohl Dependenz- als auch Phrasenstrukturen (erstere widerspiegeln sich in den Funktionen, letztere in den Knoten). Die syntaktische Struktur wird als Baum dargestellt. Wie in Abbildung 2.3 zu sehen ist, wird die linguistische Annotation auf mehreren Ebenen dargestellt. Auf der Wortebene (den Terminalknoten) sind die zu den Wortarten- (POS-Tags), Lemmata- und Morphologieinformationen kodiert. Die Nonterminalknoten sind mit Phrasenkategorien beschriftet, die Kanten mit den grammatischen Funktionen. Für die POS-Tags wurde das Stuttgart-Tübingen-Tagset (STTS, siehe Anhang A.2, S.79) verwendet. Im Tiger-Korpus sind auch sekundäre, beschriftete Kanten möglich, um z.B. koordinierte Sätze und Verbalphrasen zu annotieren (vgl. Brants et al. 2002: 25). Folgender Satz wird in Abbildung 2.3 als Tiger-Graph dargestellt:

(3) Ihm wird vorgeworfen, das Lebensrecht von Wachkomapatienten in Frage zu stellen.

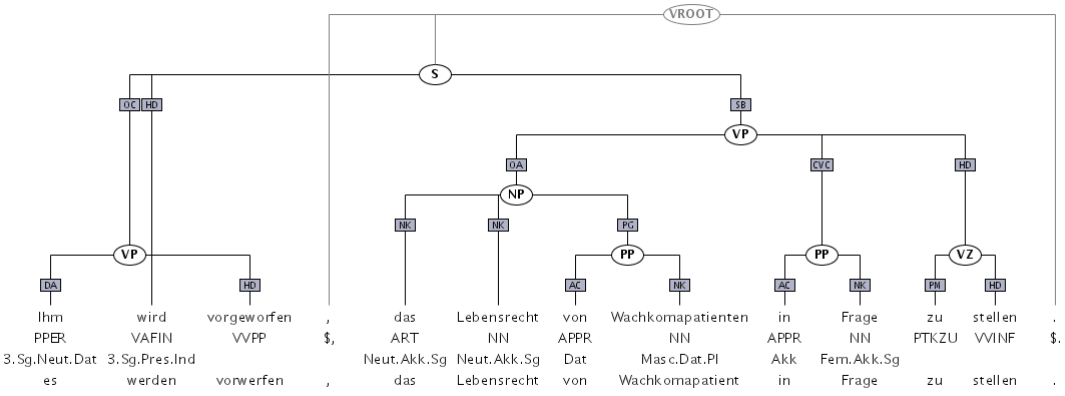


Abbildung 2.3: Beispiel für einen Graphen aus der Tiger-Baumbank

[5] http://www.ims.uni-stuttgart.de/projekte/TIGER/

Im Tigerprojekt werden verschiedene Formate für das Speichern des Korpus, den Export und die Abfrage verwendet. Die annotierten Sätze werden in Tabellen einer MySQL-Datenbank gespeichert. Die Tabellen können als ASCII-Dateien exportiert werden, um sie so besser lesbar zu machen. Aus diesem Export-Format kann eine XML-Datei in Tiger-XML erstellt werden. Der Header einer Tiger-XML-Datei enthält die Metadaten eines Korpus, wie Autor, Korpusname etc., sowie eine Deklaration der verwendeten Tags für Morphologie, POS, Nonterminal-Knoten und Kanten. Der Body enthält die linguistische Annotation in Form von gerichteten azyklischen Graphen.
Das Beispiel (3) sieht im Tiger-XML-Format so aus:

```
<s id="s26846">
 <graph root="s26846_VROOT" discontinuous="true">
   <terminals>
     <t id="s26846_1" word="Ihm" lemma="ihm" pos="PPER" morph="3.Dat.Sg.*" />
     <t id="s26846_2" word="wird" lemma="werden" pos="VAFIN" morph="3.Sg.Pres.Ind" />
     <t id="s26846_3" word="vorgeworfen" lemma="vorwerfen" pos="VVPP" morph="Psp" />
     <t id="s26846_4" word="," lemma="--" pos="$," />
     <t id="s26846_5" word="das" lemma="der" pos="ART" morph="Acc.Sg.Neut" />
     <t id="s26846_6" word="Lebensrecht" lemma="Lebensrecht" pos="NN" morph="Acc.Sg.Neut" />
     <t id="s26846_7" word="von" lemma="von" pos="APPR" />
     <t id="s26846_8" word="Wachkomapatienten" lemma="Wachkomapatient" pos="NN" morph="Dat.Pl.Masc" />
     <t id="s26846_9" word="in" lemma="in" pos="APPR" />
     <t id="s26846_10" word="Frage" lemma="Frage" pos="NN" morph="Acc.Sg.Fem" />
     <t id="s26846_11" word="zu" lemma="zu" pos="PTKZU" />
     <t id="s26846_12" word="stellen" lemma="stellen" pos="VVINF" morph="Inf" />
     <t id="s26846_13" word="." lemma="--" pos="$." />
   </terminals>
   <nonterminals>
     <nt id="s26846_500" cat="VP">
       <edge label="DA" idref="s26846_1" />
       <edge label="HD" idref="s26846_3" />
     </nt>
     <nt id="s26846_501" cat="PP">
       <edge label="AC" idref="s26846_7" />
       <edge label="NK" idref="s26846_8" />
     </nt>
     ...
     <nt id="s26846_506" cat="S">
       <edge label="OC" idref="s26846_500" />
       <edge label="HD" idref="s26846_2" />
       <edge label="SB" idref="s26846_505" />
     </nt>
     <nt id="s26846_VROOT" cat="VROOT">
       ...
     </nt>
   </nonterminals>
 </graph>
</s>
```

Jedes Terminal hat zusätzlich auch noch die Attribute `case`, `number`, `gender`, `person`, `degree`, `tense` und `mood`, die aus Platzgründen im Beispiel weggelassen

wurden. Diese Attribute sind zwar redundant, da sie bereits in `morph` enthalten sind, sie vereinfachen aber den Zugriff auf die einzelnen Werte.
Baumbanken im Tiger-XML-Format können in das Such- und Abfragewerkzeug TigerSearch geladen werden (siehe Kapitel 2.1.4, S.12). So konvertiert lassen sich auch Negra- und Penn-Treebank-Bäume graphisch anzeigen.
Für die Annotation der Tiger-Baumbank wurde *Annotate*[6] benutzt, eine graphische Benutzeroberfläche, mit der das Tagging und Parsing interaktiv gestaltet wird. Das heisst, der Parser schlägt Annotationen vor, die vom menschlichen Annotator korrigiert werden. Bei einem erneuten Trainingsdurchlauf fliessen die manuellen Korrekturen in die Analyse des Parsers ein. Das Tiger-Korpus ist im Gegensatz zur Penn-Treebank für Forschungszwecke frei erhältlich.

2.1.4 Suchwerkzeuge

Um solche Baumbanken für linguistische Anfragen systematisch erforschen zu können, braucht es Suchwerkzeuge. Für die Penn-Treebank wurde `tgrep` entwickelt, für das Tiger-Korpus gibt es den graphischen TigerSearch. Diese Suchwerkzeuge werden in den nächsten Abschnitten näher vorgestellt.
Die Abfragesprache `tgrep` (bzw. `tgrep2`) ist eine Art Erweiterung des Unixtools `grep` ('**g**lobal/**r**egular **e**xpression/**p**rint'), welches der Suche und Filterung definierter Zeichenketten in Dateien dient. Anstelle von normalem Text kann man mit `tgrep2` Bäume (im Penn-Treebank-Format) durchsuchen. In der Tabelle 2.1 sind Operatoren von `tgrep` aufgelistet, welche verschiedene Dominanz- und Präzedenzrelationen beschreiben. Unter Dominanz wird die hierarchische Relation zwischen zwei Knoten verstanden, mit Präzedenz ist die lineare Abfolge gemeint.

Ausdruck	**Beschreibung**
A < B	A dominiert B direkt
A << B	A dominiert B
A <- B	B ist das letzte Kind von A
A <<, B	B ist dasjenige Kind von A, welches am weitesten links ist
A <<‘ B	B ist dasjenige Kind von A, welches am weitesten rechts ist
A . B	B folgt direkt auf A
A .. B	B folgt auf A
A $ B	A und B sind Geschwisterknoten
A $. B	A und B sind Geschwisterknoten und B folgt direkt auf A
A $.. B	A und B sind Geschwisterknoten und B folgt auf A

Tabelle 2.1: `tgrep`-Operatoren

[6] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/annotate.html`

Neben diesen Operatoren kann man auch die gebräuchlichen regulären Ausdrücke verwenden. Zum Beispiel passt `NP < PP` auf einen NP-Knoten, welcher direkt eine Präpositionalphrase dominiert. Der reguläre Ausdruck `/^NP/` passt auf jeden Knoten, dessen Namen mit `NP` beginnt, schliesst also z.B. `NP-SUBJ` mit ein. `NP<<PP.VP` passt auf eine Nominalphrase, die eine Präpositionalphrase dominiert und direkt von einer Verbalphrase gefolgt ist. Unter `http://www.ldc.upenn.edu/ldc/online/treebank/` kann (bzw. konnte) man die Penn-Treebank mit `tgrep` online abfragen. Eine detailliertere Einführung in `tgrep2` ist in Rohde (2002) zu finden.

Für das Tiger-Korpus wurde die in Java geschriebene graphische Oberfläche TigerSearch[7] entwickelt. Mit Hilfe eines Tools namens TigerRegistry lassen sich Korpora verschiedener Formate in TigerSearch hineinladen. Die Bäume der Baumbank werden dort graphisch dargestellt und können durchsucht werden. Auch die Resultate einer Suche werden als Graphen angezeigt.
Die Abfragesprache von TigerSearch besteht aus drei Ebenen, der Knotenebene, der Knotenrelationsebene und der Graphbeschreibungsebene. Auf der Knotenebene werden Knoten durch Boolesche Ausdrücke über Merkmal-Wert-Paaren beschrieben. Die gesuchten Merkmal-Wert-Paare sind im Tiger-XML bereits klar ersichtlich:

```
<t id="s26846_10" word="Frage" lemma="Frage" pos="NN" morph="Acc.Sg.Fem"/>
<nt id="s26846_504" cat="NP">
```

Beispielsweise liefert der Ausdruck `[cat="NP"]` alle Knoten mit der Kategorie NP zurück.

```
[word="zu" & pos="PTKVZ"]
```

passt auf Terminalknoten mit dem Wort *zu*, deren Part-of-Speech-Tag ein abgetrennter Verbpartikel ist. Auf der Knotenrelationsebene werden die Knoten miteinander verknüpft. Die Verknüpfungen werden ähnlich wie in `tgrep` ausgedrückt, allerdings wird für die Dominanzrelation genau das gegenteilige Zeichen benutzt, wie in der Tabelle 2.2 zu sehen ist.

[7] `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/`

Operator	**Beschreibung**	**Beispiel**
>	direkte Dominanz	`[cat="NP"] > [pos="NE"]`
>L	Dominanz mit Label L	`[cat="NP"] >NK [cat="NP"]`
>*	Dominanz	`[cat="NP"] >* [pos="NE"]`
>n	Dominanz in Distanz n (n>0)	`[cat="NP"] >2 [pos="NE"]`
>m,n	Dominanz in Distanz d ($m \leq d \leq n$)	`[cat="NP"] >2,3 [pos="NE"]`
>@l	Left Corner Relation	`[cat="NP"] >@l [word="die"]`
>@r	Right Corner Relation	`[cat="NP"] >@r [word="Jahr"]`
.	direkte Präzedenz	`[word="die"] . [word="Katze"]`
.*	Präzedenz	`[word="die"] .* [word="Katze"]`
$	Geschwisterknoten	`[word="die"] $ [cat="NP"]`
!	Negation	`[cat="NP"] !> [pos="NE"]`

Tabelle 2.2: Knotenrelationen in TigerSearch

Auf solche Ausdrücke kann mit Variablen referenziert werden. Ein Variablenname beginnt mit dem Symbol `#`. In TigerSearch können Merkmalswerte, Merkmalsbedingungen und Knoten über Variablen referenziert werden.

Beispiel für eine Variable eines Merkmalwertes (`#noun`):
`[pos = #noun] . [pos = #noun:("NN" | "NE")]`
Beispiel für eine Variable einer Merkmalsbedingung (`#f`):
`[#f:(pos="APPR")] .* [#f]`
Beispiel für eine Variable eines Knotens (`#np`):
`#np:[cat="NP"] &  #np > [pos="ADJA"] & #np > [pos="NN"]`

Auf der Graphbeschreibungsebene werden die Knotenrelationen durch Boolesche Ausdrücke miteinander verknüpft. Es stehen auch Prädikate wie `discontinuous(#phr)` (die Phrase `#phr` enthält kreuzende Kanten) und `arity(#phr,n)` (die Phrase `#phr` enthält n Töchterknoten) zur Verfügung. Mit Variablen kann auf bereits beschriebene Knoten verwiesen werden (vgl. Lezius 2001: 421):

```
#n:[cat="S"] > [pos="VVFIN"] & #n > [pos="NN"]
#n:[cat="NP"] > RC [cat="S"] & discontinous(#n)
```

Mit TigerSearch gibt es neben der rein textuellen Anfrage auch die Möglichkeit, die Anfragen graphisch zusammenzustellen.

Die Anfrage `#n1:[cat="NP"] > [word="Katze"] & #n1 > [word="die"]` ("Finde alle Knoten mit der Kategorie NP, welche die Terminalknoten mit den Wörtern *die* und *Katze* dominieren") ist in Abbildung 2.4 in der graphischen Form zu sehen.

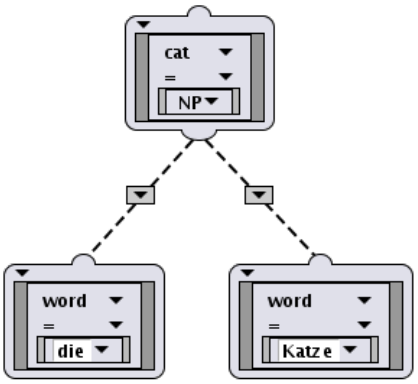


Abbildung 2.4: Graphische Anfrage in TigerSearch

Weitergehende Ausführungen zu TigerSearch und der Abfragesprache finden sich im *User's Manual* (König et al. 2003).

2.2 Parallele Baumbanken

Für (statistische) maschinelle Übersetzungssysteme werden *parallele Korpora* benötigt. Ein paralleles Korpus enthält Übersetzungen desselben Textes in mehr als eine Sprache, deren Textsegmente *aligniert* sein können. "Bei der Alignierung werden zwischen Abschnitten, Sätzen oder Wörtern in Texten in verschiedenen Sprachen (in der Regel ein Ausgangstext und eine Übersetzung davon) explizite Entsprechungen zwischen den korrespondierenden Stellen in den zwei Texten dargestellt (meist durch die Darstellung in zwei Spalten)"[8]. Der Originaltext kann im parallelen Korpus vorhanden sein, was aber nicht zwingend notwendig ist.
Vor allem auch in der computergestützten Übersetzung (CAT) wird mit alignierten parallelen Korpora gearbeitet. Dort werden alignierte Segmente in sogenannten Translation Memories gespeichert, und bei neu zu übersetzenden Sätzen kann über Musterabgleich darauf zugegriffen werden (vgl. Dorna und Jekat 2001: 569).
Bei parallelen Baumbanken werden die linguistischen Informationen und die Alignierungen vereint. Da in diesem Fall die Information über die Struktur der Sätze zur Verfügung steht, können die Alignierungen nicht nur auf Wort- und Satzebene stattfinden, sondern auch auf Phrasenebene.
In diesem Bereich wurde bisher aber nur wenig geforscht, was daran liegen mag, dass bereits der Aufwand zur Erstellung einer einzelnen Baumbank hoch ist und das Alignieren noch zusätzlichen Arbeitsaufwand bedeutet (vgl. Volk und Samuelsson 2004). An der Karls-Universität in Prag wurde die Czech-English Penn-Treebank speziell für maschinelle Übersetzung erstellt[9]. An der Universität Linköping gibt es das Projekt LinES, eine englisch-schwedische parallele Baumbank, die auf dem LTC (Linköping Translation Corpus) basiert und manuell überprüfte Satz- und Wortalignierungen enthält. LinES soll in erster Linie dazu dienen, Variationen in Übersetzungen häufiger syntaktischer Strukturen von Englisch nach Schwedisch zu untersuchen (vgl. Ahrenberg 2007). Zhechev und Way (2008) von der Dublin City University haben einen (sprachunabhängigen) Ansatz zur automatischen Generierung von parallelen Baumbanken gewählt, um grosse Datenmengen als Basis für maschinelle Übersetzungssysteme zu erhalten. In Kapitel 3.3 werde ich noch näher auf ihre Alignierungstechniken eingehen, die sich bis zu einem gewissen Grad mit meinem Ansatz zur automatischen Alignierung vergleichen lassen.

[8] Quelle: CL-Glossar, `http://www.cl.uzh.ch/kitt/clglossar/`

[9] `http://ufal.mff.cuni.cz/pcedt/doc/PCEDT_main.html`

2.2.1 Smultron

Die für diese Arbeit verwendete parallele Baumbank ist SMULTRON (Stockholm MULtilingual TReebank – *smultron* ist schwedisch und bedeutet 'Walderdbeere')[10], ein Korpus, das an der Stockholm University entstanden ist. Smultron 1.0 besteht aus ca. je 1000 getaggten und geparsten Sätzen mit etwa 18'000 Tokens in Deutsch, Englisch und Schwedisch. Die parallelen Texte stammen einerseits aus den ersten beiden Kapiteln von Jostein Gaarders *Sofies Welt* (norwegisches Original: *Sofies verden: roman om filosofiens historie*), sowie aus einer Sammlung von Wirtschaftstexten (ABB-Presseberichte, SEB-Jahresberichte, Berichte über das Bananenzertifizierungsprogramm der Rainforest Alliance). Während diese Arbeit entstand, wurde Smultron mit Handbuchtexten in Spanisch, Englisch, Deutsch und Schwedisch erweitert (vgl. Göhring 2009).

2.2.2 Erstellung der einzelnen Baumbanken für Smultron

Beim Erstellen der parallelen Baumbank wurden die monolingualen Baumbanken mit *Annotate* annotiert. In einem ersten Schritt wurden die Sätze automatisch mit POS-Tags getaggt. Für das Englische wurde das Penn-Treebank-POS-Tagset verwendet, die deutsche Baumbank wurde mit STTS annotiert und die schwedische mit einer angepassten Version des SUC-Tagsets (Stockholm-Umeå Corpus). Für den spanischen Teil wurde sowohl auf morphologischer als auch auf syntaktischer Ebene eine modifizierte Version des AnCora-Tagsets benutzt (vgl. Rios Gonzales et al. 2009: 60f). Beim semi-automatischen Parsen wurde für die englische Baumbank das Penn-Treebank-Format verwendet (siehe Kapitel 2.1.1, S.3), für die deutsche das Tiger-Annotationsschema (siehe Kapitel 2.1.3, S.9). Für die schwedische Baumbank wurde eine angepasste Version der deutschen Tiger-Guidelines verwendet (vgl. Volk und Samuelsson 2004). Die unterschiedlichen Annotationsschemata der Tiger-Baumbank und der Penn-Treebank können teilweise beim Annotieren problematisch sein, da das Tiger-Format viel flacher ist als das Penn-Treebank-Format und ersteres kreuzende Kanten erlaubt. Die Formate wurden aber so gewählt, dass die monolingualen Baumbanken mit bereits existierenden Baumbanken kompatibel und somit eigenständig benutzbar sind (vgl. Samuelsson und Volk 2006: 2).

[10] `http://www.ling.su.se/DaLi/research/smultron/index.htm`

2.2.3 Alignierung in Smultron

Für die Alignierung wurden die Baumbanken in das Tiger-XML-Format (siehe Kapitel 2.1.3, S.10) konvertiert. So werden die Baumbanken zur manuellen Alignierung in den TreeAligner eingelesen. Eine ausführliche Beschreibung des TreeAligners findet sich in Kapitel 4 (S.31 ff). Die Alignierungen werden ihrerseits in folgendem XML-Format gespeichert:

```
<?xml version="1.0" encoding="UTF-8"?>
<treealign subversion="0" version="2">
    <head>
        <alignment-metadata>
            <description>
                The first two chapters of Sophies World in English and German
            </description>
            <license>Fill in here</license>
            <author>Fill in here</author>
            <date>2008-9-10</date>
            <revision>15</revision>
            <history>
                <change date="2008-09-10" author="System">save operation</change>
                <change date="2008-09-16" author="user">save operation</change>
            </history>
        </alignment-metadata>
        <treebanks>
            <treebank id="de" filename="SMULTRON_DE_Sophies_World.xml"/>
            <treebank id="en" filename="SMULTRON_EN_Sophies_World.xml"/>
        </treebanks>
        <alignment-features>
            <alignment-feature color="red" name="fuzzy">
                Approximate translation correspondence
            </alignment-feature>
            <alignment-feature color="green" name="good">
                Precise translation correspondence
            </alignment-feature>
        </alignment-features>
       <settings>
            <display-options>
                <zoom value="1.00"/>
                <top-treebank treebank-id=""/>
            </display-options>
       </settings>
    </head>
    <alignments>
        <align comment="None" type="good">
            <node treebank_id="en" node_id="s1_1"/>
            <node treebank_id="de" node_id="s1_1"/>
        </align>
        <align comment="None" type="good">
            <node treebank_id="en" node_id="s1_2"/>
            <node treebank_id="de" node_id="s1_2"/>
        </align>
        ...
        <align comment="None" type="fuzzy">
            <node treebank_id="en" node_id="s215_520"/>
```

```
            <node treebank_id="de" node_id="s224_508"/>
        </align>
        ...
    </alignments>
</treealign>
```

Im Head sind Metadaten über die Baumbanken, Informationen zu den Alignierungstypen und Einstellungen gespeichert. Die Elemente mit dem `treebank`-Tag beinhalten Informationen über Dateinamen und die ID der Baumbank. Auf diese ID wird in den `alignments`-Elementen verwiesen. Eine Alignierung ist in einem `align`-Element als Knotenpaar gespeichert. Auf die Knoten wird im `node`-Element mit der Treebank-ID und der Knoten-ID verwiesen. Die Knoten-ID besteht ihrerseits aus der Satznummer und der Knotennummer. Das `align`-Element enthält das Attribut `type`, das den Typ der Alignierung speichert. In der aktuellen Version von Smultron wird nur zwischen *good* ('genau') und *fuzzy* ('ungefähr') unterschieden, grundsätzlich könnten in diesem XML-Format aber je nach Bedarf auch andere Alignierungstypen definiert und benutzt werden.
Für die Annotatoren wurden Richtlinien für das Alignieren erstellt (Samuelsson et al. 2007). Grundsätzlich wird zwischen zwei Alignierungsarten unterschieden: Phrasenalignierung und Wortalignierung. Unter Phrasen werden die Nonterminalknoten eines Baumes verstanden. Wortalignierung wird zwischen den Terminalknoten eines Baumpaares gemacht. Phrasen-zu-Wort-Alignierungen sind nicht vorgesehen.
Es sollten so viele Alignierungen wie möglich gemacht werden. Das Ziel ist es, sinngemässe Entsprechungen der Übersetzungen zu kennzeichnen. Sinngemäss meint in diesem Zusammenhang, dass die Textsegmente auch ausserhalb des aktuellen Kontextes als Übersetzungseinheiten ('*translation units*') für ein maschinelles Übersetzungssystem dienen könnten. Mehrfachalignierungen können vorkommen. Einerseits sind m:n Satzalignierungen möglich, andererseits sind auch 1-zu-n-Wortalignierungen erlaubt. So wird z.B. *Jahreszeit* mit *time*, mit *of* und mit *year* aligniert.

3 Automatische Alignierung

Statistische maschinelle Übersetzungsverfahren benötigen grosse Mengen an parallelen Texten, die auf Satz- und Wortebene aligniert sind. Bei diesen Textmengen sind manuelle Alignierungen nicht machbar; es müssen automatische Verfahren angewandt werden, um diese Alignierungen zu erstellen. Im Folgenden werde ich die üblichen Techniken zur automatischen Alignierung näher beschreiben.

3.1 Satzalignierung

Obwohl Satzalignierung für dieses Projekt nicht berücksichtigt wurde, will ich doch hier kurz auf Techniken der automatischen Satzalignierung eingehen.
Bevor man mit der Wortalignierung beginnen kann, müssen erst die übergeordneten Segmente wie Paragraphen und Sätze aligniert werden, um die Wortalignierung eingrenzen zu können. Paragraphen sind verhältnismässig leicht zu alignieren, da die Dokumentstrukturen meistens übereinstimmen und höchstens kurze Abschnitte in den Übersetzungen nicht immer übernommen werden. Sie werden aufgrund ihrer Länge oder Formatierung aligniert.
Sätze können nicht nur 1:1 aligniert werden, sondern es kann durchaus auch vorkommen, dass ein Satz aus der Quellsprache mit zwei Sätzen aus der Zielsprache übersetzt wird, oder dass ein Satz der Zielsprache keine Entsprechung in der Quellsprache hat. Empirisch gesehen werden Sätze, zumindest in den untersuchten Textsorten, vorwiegend 1:1 aligniert. 1:2 Alignierungen schon viel weniger häufig und 1:n, mit n>2 Alignierungen sehr selten (vgl. Kay und Röscheisen 1993: 124). Beispiele für Satzalignierungen sind in der Tabelle 3.1 aufgeführt.

	Satz	Englisch	Satz	Deutsch
1:1	11	There were no other houses beyond her garden, which made it seem as if her house lay at the end of the world.	11	Ihr Haus schien am Ende der Welt zu liegen, denn hinter ihrem Garten gab es keine weiteren Häuser mehr, nur noch Wald.
1:0	12	This was where the woods began.	–	–
1:2	27	The white envelope read: "Sophie Amundsen, 3 Clover Close."	28/29	"Sofie Amundsen", stand auf dem kleinen Briefumschlag. "Kløverveien 3 ".

Tabelle 3.1: Beispiele für (manuelle) Satzalignierungen aus Smultrons *Sofies Welt*

Die meisten Lösungsansätze verwenden entweder auf Satzlänge basierende Modelle, beziehen Wörterbücher mit ein, oder kombinieren beide Methoden. Die Methode der längenbasierten Satzalignierung geht auf Gale und Church (1993) zurück. Sie gehen davon aus, dass die Zeichenlänge von sich entprechenden Sätzen korreliert. Lange Sätze in der Quellsprache werden also tendenziell in lange Sätze in der Zielsprache übersetzt, kurze Sätze in der Quellsprache entsprechen auch kurzen Sätzen in der Zielsprache. Der Ansatz von Gale und Church basiert auf einem statistischen Modell für die Distanz zwischen zwei Sätzen (einem statistischen Modell von Zeichenlängen) und einer dynamischen Programmiermethode[1]. In ihrer Evaluation der Alignierungen für Englisch-Französisch und Englisch-Deutsch zeigen sie, dass der Algorithmus vor allem für 1-zu-1-Alignierungen sehr gut funktioniert. Dieser Ansatz wurde seither auch auf viele verschiedene Sprachpaare angewandt und erweist sich für die meisten als sehr präzise. In gewissen Ansätzen wurde die Satzlänge über die Anzahl der Wörter anstatt über Zeichenanzahl ermittelt. Diese Methode liefert jedoch schlechtere Resultate als zeichenbasierte Modelle (vgl. Tiedemann 2003: 9f.).

Die wortbasierte Satzalignierung wurde von Kay und Röscheisen (1993) eingeführt. Diese basiert auf einem iterativen Algorithmus, der mit Ankerwörtern arbeitet. Als Startwerte werden die ersten und letzten paar Sätze der parallelen Texte als Kandidaten für eine Alignierung angenommen. In einem zweiten Schritt werden in den Sätzen, die als Kandidaten vorgemerkt sind, Wörter mit gleicher

[1]Dynamische Programmierung ist ein Paradigma zum algorithmischen Lösen von Optimierungsproblemen. Das Verfahren der dynamischen Programmierung besteht darin, zuerst die optimalen Lösungen der kleinsten Teilprobleme direkt zu berechnen und diese dann geeignet zu einer Lösung eines nächstgrößeren Teilproblems zusammenzusetzen. (Quelle: Wikipedia)

bzw. ähnlicher Verteilung gesucht. Unter Wörtern werden in diesem Ansatz Token verstanden. Kommen die Wortpaare häufig genug vor, werden sie in die Tabelle der Wortalignierungen aufgenommen. Die Satzpaare, in denen die häufigsten Wortpaare vorkommen, werden in einer neuen Tabelle gespeichert, einmal pro Wortpaar, d.h. wenn mehrere Wortpaare in einem Satzpaar vorkommen, wird dieses Satzpaar mehrmals hinzugefügt. Häufig vorkommende Satzpaare werden schlussendlich wieder zu den Kandidaten hinzugefügt, und der Zyklus beginnt mit der Suche nach neuen Wortpaaren erneut.
Die verschiedenen Ansätze lassen sich natürlich auch kombinieren. Mit Techniken zur automatischen Satzalignierung können über 90% Genauigkeit erreicht werden. Weitere Verbesserungen sind vor allem in schwierigen Fällen von "verrauschten" Korpora mit unvollständigen oder divergenten Übersetzungen möglich und wünschenswert (vgl. Tiedemann 2003: 11).

3.2 Wortalignierung

Sich entsprechende Wörter von parallelen Korpora miteinander in Verbindung zu setzen, nennt man *Wortalignierung*. Alle statistischen Übersetzungsmodelle basieren auf der Idee der Wortalignierung. In Abbildung 3.1 ist ein Beispiel für Wortalignierung zu sehen. Die grün gestrichelten Linien stellen genaue Übersetzungen dar, die roten ungefähre Übersetzungen (*good* vs *fuzzy*).

Abbildung 3.1: Beispiel für Wortalignierung

Im Gegensatz zur automatischen Satzalignierung ist Wortalignierung eine weitaus anspruchsvollere Aufgabe, denn weder lassen sich die Wortlängen so einfach wie Satzlängen aufeinander abbilden[2], noch kann man sich an der Wortstellung orientieren, da diese häufig unterschiedlich ist. Auch sind Wortgrenzen ("Wort" im Sinne von lexikalischen Einheiten) viel schwieriger zu definieren als Satzgrenzen. Allerdings können einige in der Satzalignierung angewandte Techniken auch für die Wortali-

[2]Bei verwandten Sprachen funktioniert das tendenziell zwar, und Ähnlichkeitsmasse sind ein Teil der Technik zur Wortalignierung, doch es lässt sich nicht als einziges Mass verwenden.

gnierung übernommen werden. Ankerpunkte finden ist im Grunde nichts anderes, als Wörter zu alignieren.
Meistens wird bei der Wortalignierung eine vollständige Alignierung aller lexikalischen Einheiten im Korpus angestrebt. Dies führt dann oftmals aufgrund von lexikalischen, strukturellen und grammatikalischen Unterschieden, paraphrasierten Übersetzungen etc. zu ungenauen (engl. *fuzzy*) Übersetzungsrelationen (vgl. Tiedemann 2003: 11f). Es gibt hauptsächlich zwei Methoden zur automatischen Wortalignierung, welche von Tiedemann als *Assoziationsansätze* (engl. *association approaches*) und Schätzungsansätze (engl. *estimation approaches*) bezeichnet werden. Der Schätzungsansatz wird auch statistische Alignierung genannt. Da allerdings beide Ansätze eine Art Statistik verwenden, ist dieser Begriff unscharf.

3.2.1 Assoziationsansatz

Um Wörter mit Hilfe des Assoziationsansatzes zu alignieren, müssen folgende Schritte ausgeführt werden:

Lexikalische Segmentation: Zuerst müssen die Wortgrenzen bzw. die Grenzen der lexikalischen Einheiten in beiden Sprachen identifiziert werden. Lexikalische Einheiten können Phrasen oder bisweilen ganze Satzteile beinhalten. Es ist möglich, dass je nach Sprachpaar diese Grenzen anders gesetzt werden müssen.

Übereinstimmung: Mit Hilfe eines sogenannten *Assoziationswörterbuchs*, welches gewichtete Einträge enthält, werden die möglichen Übersetzungsrelationen zwischen lexikalischen Einheiten ermittelt.

Alignierung und Extraktion: Die sichersten Übersetzungen werden als Alignierungen gekennzeichnet. Aus den Alignierungen kann ein zweisprachiges Wörterbuch extrahiert werden.

Um die Wahrscheinlichkeit des Auftretens eines Wortes im Quellsatz und eines Wortes im Zielsatz zu bestimmen, werden Kookkurrenzmasse wie der *t-score* oder der *Dice-Koeffizient* benutzt. Andere empirische Alignierungstechniken basieren auf Zeichenketten-Ähnlichkeitsmassen (engl. *string similarity measures*). Vor allem für eng verwandte Sprachen lassen sich mit solchen Ähnlichkeitsmassen häufig sogenannte *cognates* finden, also etymologisch verwandte Wörter.
Natürlich gilt für Wörter noch mehr als für Sätze, dass sie nicht nur 1:1 sondern 1:n aligniert werden können. Zu solchen Mehrwortausdrücken (engl. *multi word units* – MWUs) gehören strukturelle und konzeptuelle Einheiten wie komplexe Nominalphrasen (Beispiel: *die Behauptung, dass er kommt*), *phrasal verbs* z.B. im Englischen (*look up*, *look for*), idiomatische Ausdrücke sowie andere phrasale Konstrukte.

In einigen Ansätzen werden Mehrwortausdrücke vor der Alignierung bestimmt, z.B. mit Hilfe von Techniken zur Identifikation von Kollokationen, in anderen Ansätzen werden sie dynamisch während dem Alignierungsprozess ermittelt (vgl. Tiedemann 2003: 18f).

3.2.2 Schätzungsansatz

Häufig wird Wortalignierung mit Hilfe von probabilistischen Alignierungsmodellen erstellt. Diese Modelle werden aus grossen parallelen Korpora berechnet. Die meisten Übersetzungsmodelle, die in der statistischen maschinellen Übersetzung benutzt werden, gehen auf die fünf IBM-Modelle zurück, die in Brown et al. (1993) beschrieben sind. Die zugrunde liegenden mathematischen Modelle stammen ursprünglich aus der Informationstheorie. Dabei wird angenommen, dass die Übersetzung nichts anderes ist als die Übertragung einer Nachricht, die vom Sender in der Quellsprache **s** verschickt und durch Übertragungsfehler, dem sogenannten "Signalrauschen", beim Empfänger in der Zielsprache **t** ankommt (Noisy-Channel-Modell[3]) (vgl. Dorna und Jekat 2001: 568f). Eine Alignierung wird üblicherweise als eine Folge von verborgenen Verbindungen zwischen den Wörtern der Zielsprache und den Wörtern der Quellsprache modelliert. Die Beziehung zwischen Alignierung **a** und Übersetzung kann folgendermassen ausgedrückt werden:

$$P(\mathbf{t}|\mathbf{s}) = \sum_{a} P(\mathbf{t}, \mathbf{a}|\mathbf{s})$$

Das Übersetzungsmodell berücksichtigt jetzt also alle möglichen Zuordnungen von Wörtern der Quellsprache zu Wörtern der Zielsprache. Um jeden Kontext abzudecken, in der eine Wortalignierung auftreten kann, wird dieser Term auf Wortebene aufgeschlüsselt, wobei der Quellstring $\mathbf{s} = s^L = s_0 s_1 s_2 .. s_L$ von L Wörtern der Quellsprache und dessen Übersetzung $\mathbf{t} = t^M = t_0 t_1 t_2 .. t_M$ von M Wörtern der Zielsprache sind. Eine Alignierung ist eine Folge von M Verbindungen der Form $\mathbf{a} = a^M = a_0 a_1 a_2 .. a_M$, wobei $a_m \in 0, .., L$. Dieses Alignierungsmodell wird auch *direktional* genannt, weil es nicht symmetrisch ist. Es erlaubt nicht mehrere Verbindungen von einem Wort der Zielsprache zu Wörtern in der Quellsprache. Dieses Problem lässt sich jedoch über Mehrwortausdrücke lösen (d.h. die Mehrwortausdrücke werden zuerst bestimmt und für die Berechnung der Alignierungen als eine Einheit, bzw. als ein Wort, betrachtet). Für die Wortebene sieht die Gleichung nun also wie folgt aus:

[3]Dieses Noisy-Channel-Modell wurde ursprünglich für effektive Nachrichtenkanäle mit Rauschen (wie Telefon oder Funk) entwickelt.

$$P(t^M|s^L) = \sum_{a^M} P(t^M, a^M|s^L) = P(M|s^L) \sum_{a^M} \prod_{m=1}^{M} P(t_m, a_m|t^{m-1}, a^{m-1}, s^L)$$

Das heisst, bei gegebenem Quellsprache-Satz kann die bedingte Wahrscheinlichkeit des Satzes der Zielsprache und seiner Alignierung als das Produkt aller Zielsprachewörter t_m und deren Alignierungen a_m ausgedrückt werden, wenn die vorangehenden Wörter t^{m-1}, deren Alignierungen a^{m-1} und der Satz der Quellsprache s^L gegeben sind. Die Summe aller bedingten Wahrscheinlichkeiten wird mit der Wahrscheinlichkeit der Länge des Zielsprachesatzes, unter der Voraussetzung vom Satz der Quellsprache, $P(M|s^L)$, multipliziert (vgl. Tiedemann 2003: 21f). Durch die hohe Anzahl der Parameter im rechten Teil der Gleichung ist die praktische Berechnung sehr komplex. Deshalb wurden verschiedene Vereinfachungen an diesem Modell vorgenommen – die sogenannten IBM-Modelle (Brown et al. 1993). Die Idee hinter den fünf Modellen ist, mit dem einfachen Modell zu beginnen und dessen Ausgabe dann für das nächst komplexere Modell zu verwenden. Das IBM-Modell 1 ist ein einfaches Wortübersetzungsmodell, das hauptsächlich von Kookkurrenzen sich entsprechender Wörter in alignierten Sätzen Gebrauch macht und annimmt, dass alle Alignierungen gleich wahrscheinlich sind. Mit dem Modell 2 wird die Wortstellung berücksichtigt, indem Positionsparameter eingeführt werden. Trainiert wird das IBM-Modell mit dem EM-Algorithmus (Expectation-Maximization-Algorithmus). In den Modellen 3-5 wird noch die Wahrscheinlichkeit miteinbezogen, mit der Quellsprachewörter zu einer bestimmten Anzahl entsprechender Zielsprachewörter aligniert werden. Die drei letzten Modelle unterscheiden sich in der Berechnung dieser Wahrscheinlichkeiten.

Wie bereits erwähnt, benutzen die meisten Übersetzungsmodelle eine Abwandlung der IBM-Modelle. Eine dieser Varianten, die in Vogel et al. (1996) beschrieben ist, benutzt Hidden Markov Models (HMMs). Dabei ist die Alignierungswahrscheinlichkeit nicht von der absoluten Position des Wortes der Übersetzung, sondern von der relativen Position abhängig. Ein HMM-basiertes Übersetzungsmodell besteht aus einem Alignierungsmodell mit den verborgenen Alignierungsparametern $P(a_m|a_{m-1})$ und einem lexikalischen Modell, das die lexikalischen Übersetzungswahrscheinlichkeiten $P(t_m|s_{a_m})$ zur Verfügung stellt.

3.3 Phrasenalignierung

Phrasenalignierung kann als weitere Informations-Ebene über der Syntaxstruktur betrachtet werden. Damit wird gekennzeichnet, welche Satzteile der einen Sprache welchen Satzteilen der anderen Sprache entsprechen. Oftmals kann man die einzelnen Wörter nicht alignieren, wohl aber die übergeordnete Phrasenstruktur. In manchen Kontexten wird unter Phrasenalignierung auch die Alignierung von Wort-N-Grammen verstanden, welche nicht unbedingt linguistischen Phrasen entsprechen. Automatische Alignierung von (linguistischen) Phrasen wird nicht sehr oft gemacht, da dazu parallele Baumbanken benötigt werden.
Für Smultron (siehe Kapitel 2.2.1) wurden für den schwedisch-englischen Sophie-Teil automatisch Phrasenalignierungen generiert (vgl. Samuelsson und Volk 2007). In einer ersten Phase wurden die SMT-Trainingsmodule Pharaoh[4] und Thot[5] benutzt, um N-Gramme zu generieren, später wurde dafür auch Moses[6] verwendet. Ein Eintrag in einer Phrasentabelle sieht z.B. so aus:

```
Der Garten Eden ||| THE GARDEN OF EDEN ||| (0) (1) (2,3) ||| (0) (1) (2)
(2) |||1 0.375 1 0.001443 2.718
```

Der Eintrag ist ein Beispiel aus einer von Moses generierten Phrasentabelle. Die ersten beiden Einträge sind die deutschen und englischen Wortfolgen. Die Klammern im dritten Feld stellen dar, mit welchen Wörtern der englischen Phrase die deutschen Wörter aligniert werden, bzw. im vierten Feld umgekehrt welche englischen Wörter mit welchen deutschen Wörtern aligniert werden: Das erste Wort in der deutschen Phrase wird mit dem Wort an Position 0 der englischen Phrase aligniert und umgekehrt (also *der* mit *the*). Auch die Wörter an Position 1 der deutschen und englischen Phrase werden aligniert. Das dritte Wort der deutschen Phrase (an Position 2), wird sowohl mit den englischen Wörtern der Position 2 und 3 aligniert. Die fünf Werte am Schluss des Eintrags beziehen sich in dieser Reihenfolge auf die Phrasen-Übersetzungswahrscheinlichkeit $\varphi(f|e)$, die lexikalische Gewichtung $lex(f|e)$, Phrasen-Übersetzungswahrscheinlichkeit $\varphi(e|f)$, die lexikalische Gewichtung $lex(e|f)$ und eine sogenannte Strafe (*phrase penalty*, immer $\exp(1) = 2.718$); f steht für *foreign*, in diesem Fall für Deutsch, e für Englisch.

In einer zweiten Phase dienten die Ausgabe der Trainingsmodule (eine Phrasen-

[4] http://www.isi.edu/licensed-sw/pharaoh/

[5] http://thot.sourceforge.net/

[6] http://www.statmt.org/moses

Tabelle) und die Baumbanken als Eingabe für ein linguistisches Alignierungsfilterprogramm, welches dann die Alignierungen zwischen den syntaktischen Phrasen der parallelen Baumbank erstellte. Das Filterprogramm extrahiert alle Phrasen einer Baumbank und deren dominierten Terminals und vergleicht diese mit den N-Grammen aus der Phrasentabelle. Z.B. wird eine Phrase in folgendem Format aus der deutschen Baumbank extrahiert: `DE: Der Garten Eden - NP s1_500`
Wenn die Wortfolge der N-Gramme mit derjenigen einer Phrase übereinstimmt, wird die Knotennummer zusammen mit dem alignierten N-Gramm und dessen Wahrscheinlichkeit gespeichert. Die Wortfolgen der Phrasen der zweiten Baumbank werden anschliessend nur noch mit den im vorangehenden Schritt gespeicherten Alignierungs-N-Grammen verglichen. Bei einer Übereinstimmung, d.h. wenn `EN: THE GARDEN OF EDEN - NP s1_500` in der Phrasentabelle gefunden wird, wird die Alignierung gespeichert: `Align DE s1_500 - EN s1_500`. So bleiben von den errechneten N-Gramm-Alignierungen aus der ersten Phase nur noch diejenigen übrig, die auch tatsächlich Alignierungen von linguistischen Phrasen entsprechen. Bei der Evaluation mit dem Goldstandard lag der $F_{0.5}$-Score[7] bei ca. 65%.
In Groves et al. (2004) wird im Hinblick auf datenorientierte Übersetzung ein mehr oder weniger sprachunabhängiger Ansatz zur automatischen Alignierung von Phrasen vorgestellt. Deren Ausgangslage ist eine englisch-französische parallele Baumbank aus dem HomeCentre-Korpus mit (LFG-)Phrasenstrukturbäumen. Das HomeCentre-Korpus enthält 810 alignierte Satzpaare eines Xerox-Drucker-Handbuchs. Da der Algorithmus zur Bestimmung der Phrasenalignierungen sehr genau beschrieben ist, werde ich hier auf den Artikel detaillierter eingehen.
Als erster Schritt werden mit Hilfe eines automatisch generierten Lexikons Wortalignierungen zwischen den Baumpaaren gesucht. Aufgrund dieser alignierten Terminalknoten wird danach bottom-up vorgegangen, einerseits mit Abgleich der Knotenlabels, andererseits mit struktureller Information. Nach jedem Schritt werden die neuen alignierten Knotenpaare gespeichert und schränken somit auch die Wahl bei der Suche nach Alignierungen der noch nicht alignierten Knoten ein. Der Algorithmus besteht aus den folgenden fünf Schritten, die für jedes alignierte Knotenpaar in der Liste durchgeführt werden.

Verb- und Objekt-Alignierung (Abbildung 3.2): Sowohl s als auch t des alignierten (Terminal-) Knotenpaares $\langle s, t \rangle$ sind Verben (die Modalverben *can* resp. *pouvez* in der Abbildung), ihre Elternknoten sind Verbalphrasen. Als weitere Bedingungen müssen sie die äussersten Kindknoten in ihren Teilbäumen sein und dieselbe Anzahl an Geschwisterknoten mit sich entsprechenden syntaktischen Labeln aufweisen. In diesem Fall werden die Geschwisterknoten von

[7]Beim $F_{0.5}$-Score wird die Präzision im Vergleich zur Ausbeute doppelt gewichtet.

s und t miteinander aligniert, was der Alignierung des Objekts des Verbs s und demjenigen des Verbs t entsprechen sollte. Die Elternknoten werden ebenfalls aligniert. In der Abbildung 3.2 sind die aus dieser Regel resultierenden Alignierungen als gestrichelte Linien dargestellt.

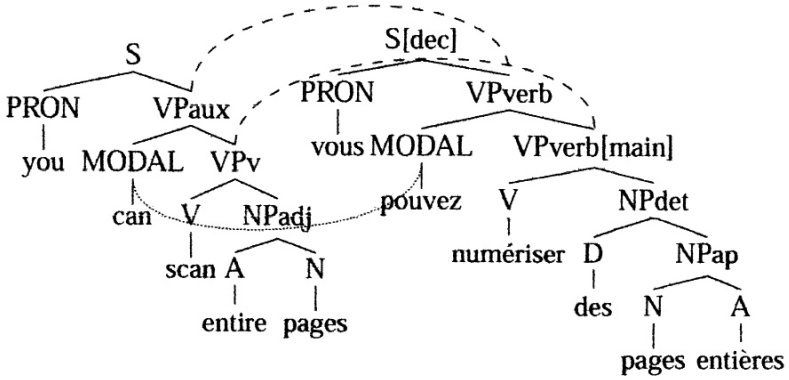


Abbildung 3.2: Verb- und Objekt-Alignierung

Elternknoten-Alignierung (Abbildung 3.3): Wenn die Geschwisterknoten der Knoten s und t eines alignierten Knotenpaares $\langle s, t\rangle$ ebenfalls aligniert sind, wird der Elternknoten par_s mit dem Elternknoten par_t verknüpft. Wenn s und t beide nur je einen nicht-alignierten Geschwisterknoten haben, werden diese beiden Geschwister auch miteinander aligniert, was im Beispiel der Abbildung der Fall ist. Dort entsprechen s dem Knoten *color* und t dem Knoten *couleur*.

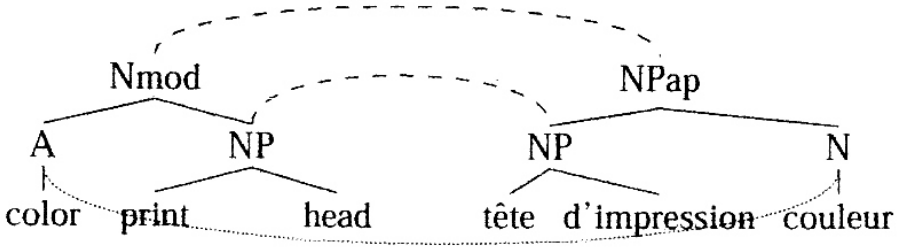


Abbildung 3.3: Elternknoten-Alignierung

NP/VP-Alignierung (Abbildung 3.4): Die Knoten s und t des alignierten Knotenpaares $\langle s, t\rangle$ sind beides Nomen (*document*/*document* im Beispiel). Die im Baum am höchsten stehenden NPs (np_s und np_t), die jeweils s und t dominieren (die Nominalphrasen, zu welchen die Pfeile in der Abbildung zeigen), werden aligniert. Dann wird von np_s und np_t je der am weitesten links liegende, direkte Kindknoten ermittelt (k_s, k_t: Im Beispiel sind das die Elternknoten der beiden unbestimmten Artikel *a* und *un*). Wenn die Label von k_s und k_t

sich entsprechen, werden auch sie aligniert. So kann der Wirkungsbereich von Nominalphrasen-Modifikatoren beibehalten werden. Auch bei zwei alignierten Verben werden die jeweils obersten dominierenden VP-Knoten miteinander aligniert.

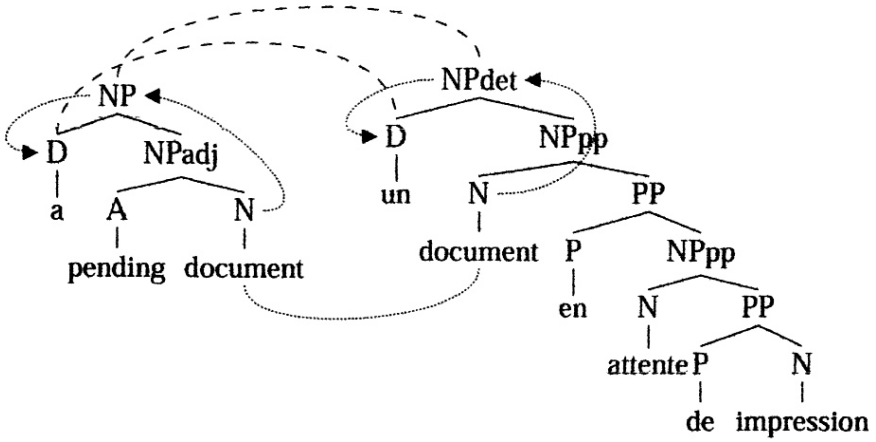


Abbildung 3.4: NP-Alignierung

Kind-Alignierung (Abbildung 3.5): Analog zur Alignierung von Elternknoten werden die direkten Kindknoten vom Knotenpaar $\langle s, t \rangle$ miteinander aligniert, wenn sie in derselben Anzahl vorhanden sind und ein übereinstimmendes Knotenlabel aufweisen. Im Beispiel sind das die Kindknoten `NP` und `VPaux` des englischen Satzknotens, bzw. `NPdet` und `VPcop` des französischen Satzknotens.

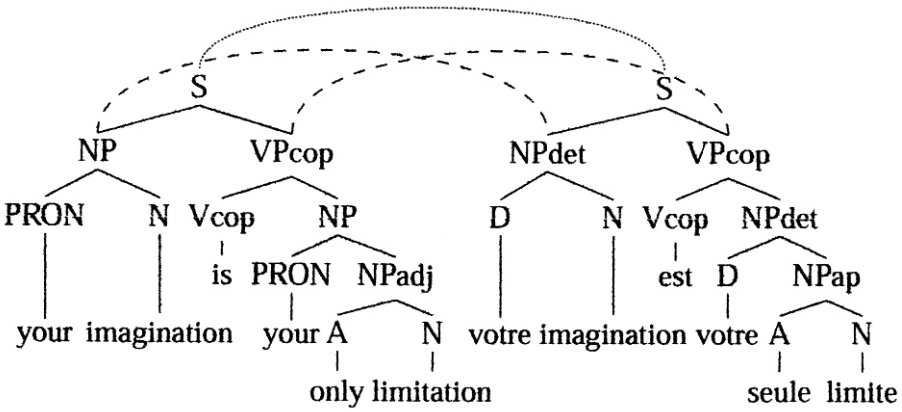


Abbildung 3.5: Kind-Alignierung

Teilbaum-Alignierung: Wenn die Teilstrukturen mit s und t des alignierten Knotenpaares $\langle s, t \rangle$ als Wurzeln isomorph sind, also genau übereinstimmende Strukturen und Labels aufweisen, werden die Knoten dieser Teilbäume aligniert.

Die Autoren gehen nicht genauer darauf ein, wie die Strukturen und/oder Labels der beiden Sprachen als übereinstimmend definiert werden. Obwohl der Algorithmus sprachunabhängig sein soll, ergibt sich der Eindruck, dass er zumindest nicht Baumbanken-unabhängig ist, denn die Übereinstimmungen müssen wohl je nach Baumstrukturen bzw. -labels wieder neu definiert werden.

Die automatischen Alignierungen wurden über einem manuell alignierten Goldstandard evaluiert. Das Resultat war ein F-Mass von 0.7064. Da das Ziel der Alignierung datenorientierte Übersetzung war, wurde auch die Übersetzungsqualität mit den automatisch generierten Phrasenalignierungen überprüft, indem die Autoren dasselbe Experiment wie Hearne und Way (2003) durchgeführt und die dort benutzten manuellen Alignierungen durch die automatisch generierten ersetzt haben. Je nach Strukturtiefe der Phrasen sind das F-Mass und der Bleu-Score der Übersetzung mit den automatischen Alignierungen nicht viel schlechter, bzw. teilweise sogar besser als diejenigen mit manuellen Alignierungen.

Eine weitere Methode zur sprachunabhängigen Alignierung von Phrasen schlagen Tinsley et al. (2007) vor (vgl. auch Zhechev und Way (2008)). Wie Groves et al. (2004) verwenden sie als Testkorpus die englisch-französische HomeCentre-Baumbank. Die Anforderungen an den Algorithmus sind folgende: Er soll unabhängig vom Sprachpaar und von den Konstituentenlabels sein, die gegebenen Baumstrukturen sollen bewahrt werden, es sollen möglichst wenige externe Ressourcen verwendet werden, und die Wort-Alignierungen sollen nicht von vornherein fix sein. Die einzige externe Ressource, die verwendet wird, ist der Moses-Dekodierer, um die Wortalignierungs-Wahrscheinlichkeiten zu berechnen. Diese Wahrscheinlichkeiten dienen als Grundlage für die Phrasenalignierung, aufgrund derer dann wieder Rückschlüsse auf die Wortalignierungen gezogen werden. Alignierungen müssen folgende Wohlgeformtheitskriterien erfüllen, um als Kandidaten in Betracht gezogen zu werden:

1. Ein Knoten kann nur einmal aligniert werden.
2. Kindknoten eines alignierten Knotens der Quellsprache können nur mit Kindknoten des damit alignierten Knotens in der Zielsprache aligniert werden. Das gilt analog auch für die Elternknoten.

Für die Knotenpaare $\langle s, t\rangle$ jedes Baumpaares $\langle S, T\rangle$ werden Alignierungshypothesen aufgestellt, die Punktzahlen $\gamma(\langle s, t, \rangle)$ zugewiesen bekommen. Alle Hypothesen mit einer Punktzahl von null werden von Anfang an ausgeschlossen. Die Punktzahlen werden über die Wortalignierungs-Wahrscheinlichkeiten berechnet. Danach werden

iterativ die Alignierungshypothesen mit den höchsten Punktzahlen ausgewählt und Hypothesen, die ersteren widersprechen, werden blockiert. Weitere Strategien werden angewandt, wenn zwei Hypothesen dieselbe Punktzahl haben. Evaluiert wurde wie bei Groves et al. (2004) in zweierlei Hinsicht. Die automatisch generierten Alignierungen wurden sowohl über einem Goldstandard (aus manuellen Alignierungen) evaluiert, als auch zum Training für ein datenorientiertes Übersetzungssystem benutzt. Die Ausgabe des Übersetzungssystems wurde dann mit derjenigen eines über manuellen Alignierungen trainierten Systems verglichen. Die höchste Präzision, die beim Vergleich mit dem Goldstandard erreicht wurde, betrug 0.6256 für alle Alignierungen. Wenn nur die nicht-lexikalischen Alignierungen berücksichtigt wurden, betrug der Wert 0.8424. Die höchste Ausbeute für alle Alignierungen betrug 0.8101, dafür war sie etwas tiefer (0.8002), wenn nur nicht-lexikalische Alignierungen berücksichtigt wurden. Für die zweite Evaluation wurden die BLEU-, NIST- und METEOR-Masse berechnet (für BLEU siehe Papineni et al. (2002), für NIST siehe Doddington (2002) und für METEOR siehe Banerjee und Lavie (2005)). Der höchste BLEU-Wert, der erreicht wurde, war 0.5333, der höchste NIST-Wert 6.9384 und der höchste METEOR-Wert lag bei 73.0014 %.

Dieses Kapitel hat eine kleine Übersicht über Techniken zur automatischen Alignierung gezeigt und einige Techniken etwas ausführlicher dargestellt. Zur automatischen Satz- und Wortalignierung gibt es schon sehr viele Experimente und Ansätze; die automatische Phrasenalignierung hingegen ist tendenziell weniger erforscht, was unter anderem daran liegt, dass die Ressourcen, die es dazu braucht (nämlich parallele Baumbanken), noch verhältnismässig spärlich vorhanden sind.

4 TreeAligner

Der TreeAligner ist ein Werkzeug zum Alignieren und Durchsuchen von parallelen Baumbanken. Er ist frei verfügbar[1] und steht unter der GNU GPL[2]. Wie bereits in Kapitel 2.2.3 erwähnt, wurde der TreeAligner im Hinblick auf Smultron erstellt. Wie TigerSearch (siehe Kapitel 2.1.4) ist der TreeAligner als graphische Benutzeroberfläche implementiert, doch im Gegensatz zu TigerSearch, das in Java geschrieben ist, ist er in Python programmiert (wieso dies so gehandhabt wurde, kann in Mettler (2007) nachgelesen werden). Auch die TigerSearch-Abfragesprache wurde für den TreeAligner in Python reimplementiert und erweitert. Die Suche im TreeAligner werde ich in Kapitel 4.2 (S. 35) näher beschreiben. Der TreeAligner stellt die Bäume mit Knoten- und Kantenlabels dar und zeigt auch kreuzende und sekundäre Kanten an (siehe Abbildungen 4.1 und 4.2). In Abbildung 4.1 kreuzt die Kante vom Satzknoten zum Verb jene der Nominalphrase.

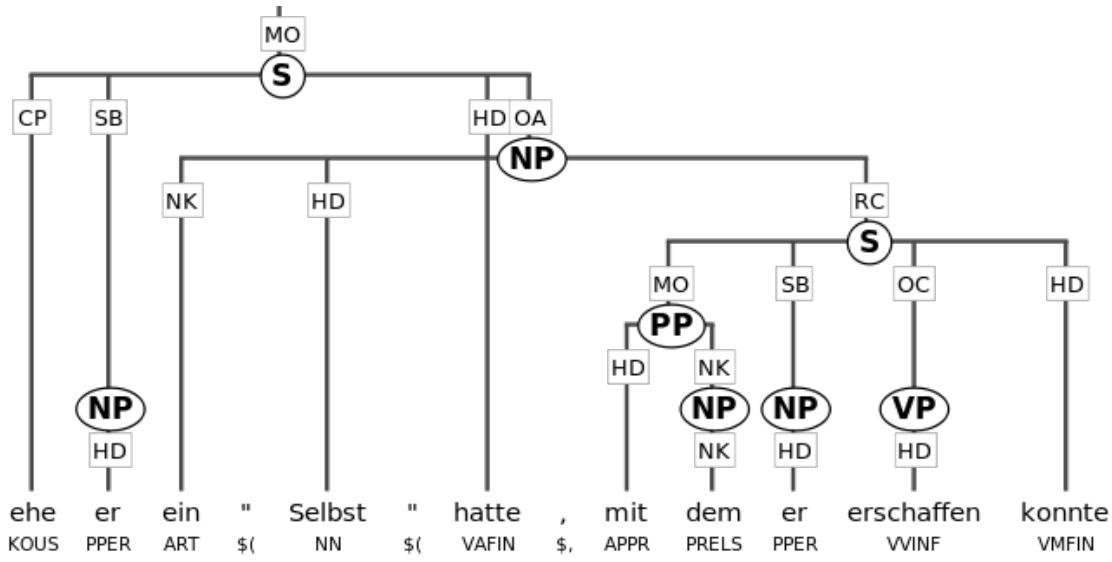


Abbildung 4.1: Beispiel einer kreuzenden Kante

[1] http://www.cl.uzh.ch/kitt/treealigner

[2] http://www.gnu.org/copyleft/gpl.html

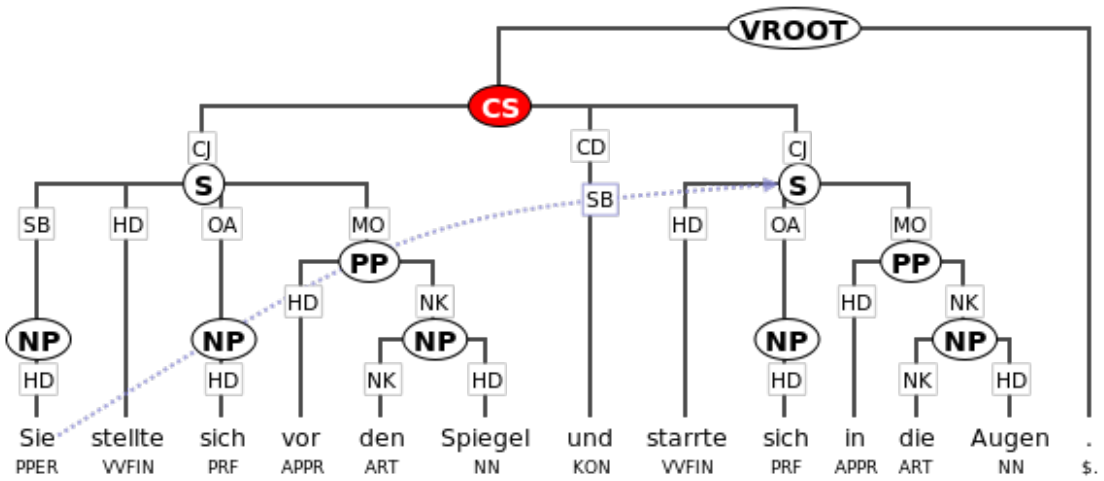


Abbildung 4.2: Beispiel einer sekundären Kante

Sekundäre Kanten ergeben sich vor allem durch Koordination, wie in Abbildung 4.2 zu sehen ist. Die sekundäre Kante in Abbildung 4.2 trägt das Kantenlabel `SB`. D.h. das Personalpronomen *Sie*, das Subjekt des ersten Teilsatzes, ist ebenfalls Subjekt des zweiten Teilsatzes.

Im folgenden Kapitel stelle ich die manuelle Alignierung und die Browsersicht im TreeAligner genauer vor.

4.1 Alignierung im TreeAligner

Eine Alignierung wird im TreeAligner erstellt, indem mit der Maus auf einen Knoten geklickt und, während die linke Maustaste gedrückt gehalten wird, kurz in eine Richtung gezogen wird. Danach wird eine Linie sichtbar, die sich mit der Maus zum Zielknoten ziehen lässt, indem man auf den Zielknoten klickt. Je nachdem, welcher Alignierungstyp eingestellt ist, wird die Linie anders eingefärbt. Standardmässig gibt es die Alignierungstypen *good* (genaue Übersetzung) und *fuzzy* (ungefähre Übersetzung). Für ein neues Projekt können aber individuelle Typen definiert werden, ebenso deren Alignierungsfarbe. Für *good* ist die Standardfarbe Grün, *fuzzy* wird in Rot angezeigt. Das Beispiel eines fertig alignierten Baumpaares ist in Abbildung 4.3 zu sehen.

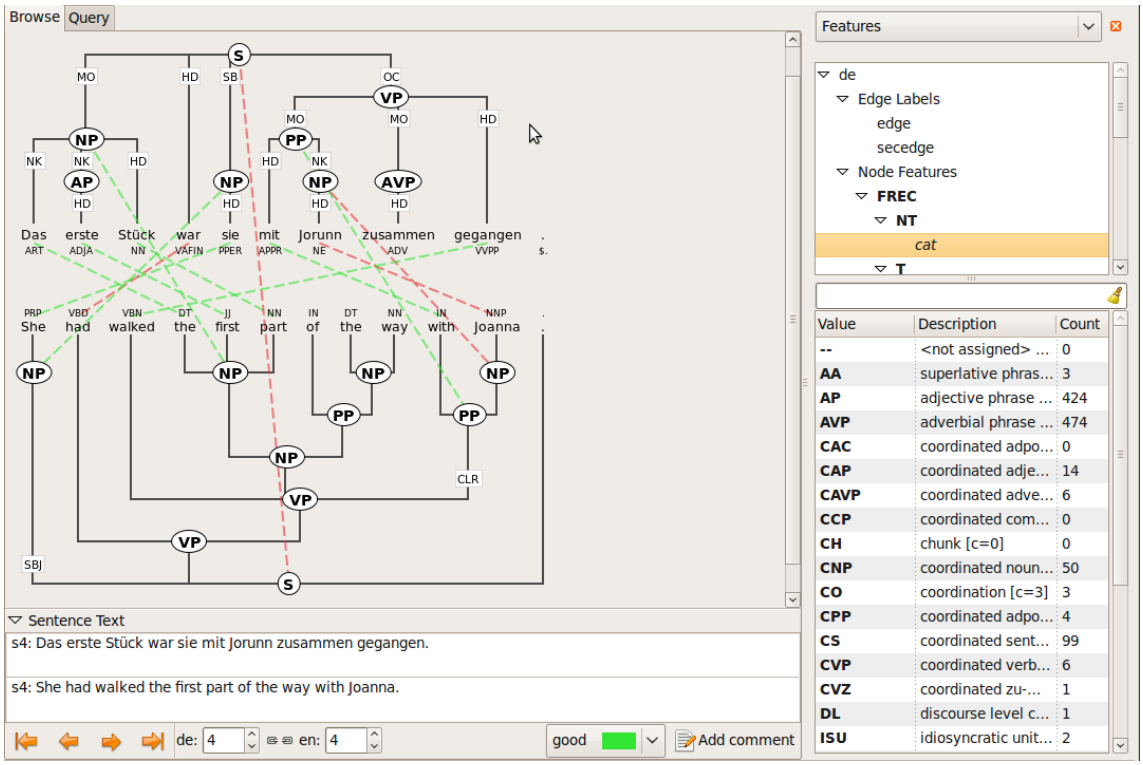


Abbildung 4.3: Beispiel eines alignierten Baumpaares

Die Information einer neuen Alignierung wird in der Projekt-XML-Datei folgendermassen gespeichert:

```
<align comment="None" type="good">
    <node treebank_id="en" node_id="s1_1"/>
    <node treebank_id="de" node_id="s1_1"/>
</align>
```

Der Knoten 1 aus dem englischen Satz 1 wurde mit dem Knoten 1 aus dem deutschen Satz 1 mit dem Typ *good* aligniert.

Mit dem Tastaturkürzel F9 lässt sich eine Seitenleiste einblenden, die weitere Informationen über die parallele Baumbank enthält: Alle alignierten Satzpaare der parallelen Baumbank, Informationen über das Projekt, die Merkmale (*features*) der Knoten- und Kantenlabels sowie die Information über allfällig inkonsistente Alignierungen im aktuellen Satzpaar. In Abbildung 4.3 sind die Merkmale zu sehen, bei denen gerade die deutschen Labels der Non-Terminal-Knoten angezeigt werden.

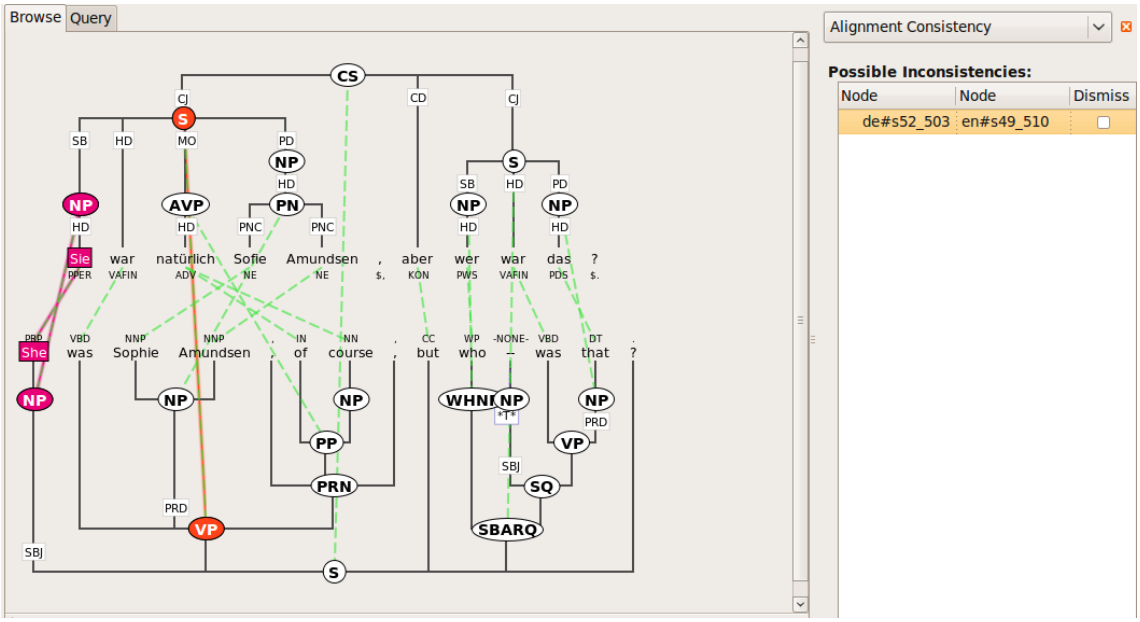


Abbildung 4.4: Mögliche Inkonsistenz

Der Inkonsistenztest wurde in der Version 1.1.90 hinzugefügt. Ein Beispiel dafür zeigt die Abbildung 4.4. Die Alignierungen der rot gefärbten Knoten sind aufgrund der Alignierungen der violett gefärbten Knoten inkonsistent. In diesem Beispiel handelt es sich um eine strukturelle Inkonsistenz, die durch die Alignierung der violett gefärbten Knoten ausgelöst wird: Das englische Pronomen, bzw. dessen dominierende Nominalphrase ist nicht in der Verbalphrase enthalten, die mit dem deutschen Teilsatz S aligniert wurde, das deutsche Pronomen, bzw. dessen dominierende Nominalphrase ist jedoch ein Kindknoten des deutschen Teilsatzes S. Darauf macht der Inkonsistenztest aufmerksam.

Für ein neues Projekt können im TreeAligner auch automatisch Sätze aligniert werden. Es wurde der Ansatz von Gale und Church (1993) implementiert, wobei der erste Schritt, die Alignierung von Paragraphen, weggelassen werden musste, da die Information über Paragraphen im Tiger-XML-Format nicht mehr vorhanden ist. Eine automatische Satzalignierung erfolgt von `VROOT` zu `VROOT` und ist im XML als `<sen-align>`-Tag erkennbar. Graphisch wird die automatische Satzalignierung blau gefärbt angezeigt, wie in Abbildung 4.5 zu sehen ist.

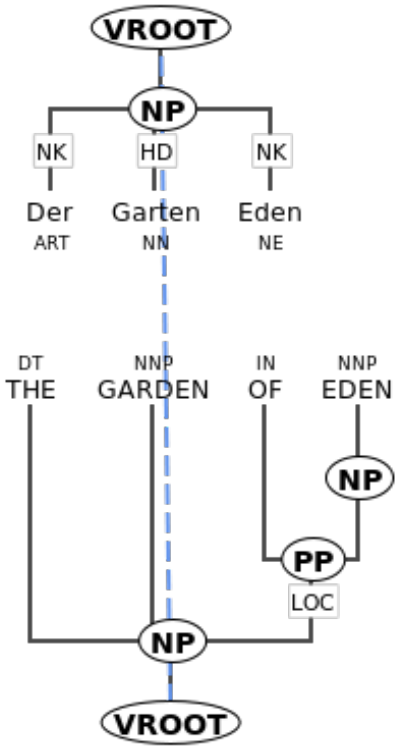


Abbildung 4.5: Automatische Satzalignierung im TreeAligner

4.2 Suche im TreeAligner

Die Suche im TreeAligner funktioniert ähnlich wie die textuelle Suche in TigerSearch (siehe Kapitel 2.1.4, S.12ff.). Die Implementation der Suchabfragesprache basiert auf Knotensets und auf bedingten kartesischen Produkten über diesen Sets (vgl. Marek et al. 2008). Die Suchabfragesprache unterscheidet sich nur in ein paar wenigen Punkten von TigerSearch. Die unspezifische Merkmalsbedingung `[]` wird im TreeAligner mit `[FREC]`[3] ausgedrückt. Für Merkmalswerte können keine Variablen gesetzt werden, also z.B. `[pos=#v:"APPR"] .* [pos=#v]` ist in der TreeAligner-Suche nicht möglich. Im Gegensatz zu TigerSearch ist es möglich, Bedingungen zu "stapeln" (gesucht ist eine Nominalphrase, welche eine Präpositionalphrase dominiert, die ihrerseits eine Nominalphrase dominiert):

- TigerSearch: `[cat="NP"] > #n:[cat="PP"] & #n > [cat="NP"]`
- TreeAligner: `[cat="NP"] > [cat="PP"] > [cat="NP"]`

Prädikate können mit dem Präfix ! negiert werden, wobei der Präfix zum Namen gehört und keinen Booleschen Operator darstellt, z.B. `!root(#x)`,

[3] `FREC` steht für alle *feature records*, d.h. sowohl für NT als auch für T.

`#np:[cat="NP"] & !arity(#n, 2)`. Vor einiger Zeit wurde die TreeAligner-Abfragesprache mit universeller Quantifizierung erweitert für Fragen wie “Finde Sätze, die keine Nominalphrasen enthalten”, welche in TigerSearch nicht implementiert ist (vgl. König et al. 2003: 39). Im TreeAligner ist es möglich, Anfragen über Mengen (‘set’) von Knoten zu stellen. Folgende Anfrage gibt alle Sätze, die keine Nominalphrasen enthalten, zurück: `#s:[cat="S"] !>* %p:[cat="NP"]`
Die genauere Beschreibung der universellen Quantifizierung im TreeAligner und deren Implementierung lässt sich in Marek et al. (2008) nachlesen.
Für beide Baumbanken kann in den oberen beiden Feldern die gesuchte Struktur im Format der Abfragesprache eingegeben werden (siehe Abbildung 4.6). Das unterste Feld spezifiziert, welche Knoten der ersten Suchanfrage mit welchen Knoten der zweiten Suchanfrage aligniert sind. Dazu werden die Variablen der Knoten mit dem Alignierungsoperator `--` verknüpft, wie im Screenshot der Abbildung 4.6 zu sehen ist. Gesucht wird ein Knoten `t1` in der deutschen Baumbank, der eine Präpositionalphrase dominiert und mit einem Knoten `t2` aus der englischen Baumbank, der seinerseits eine Präpositionalphrase dominiert, aligniert ist. Die für die Suchanfrage relevanten Knoten werden im Suchresultat rot hervorgehoben.

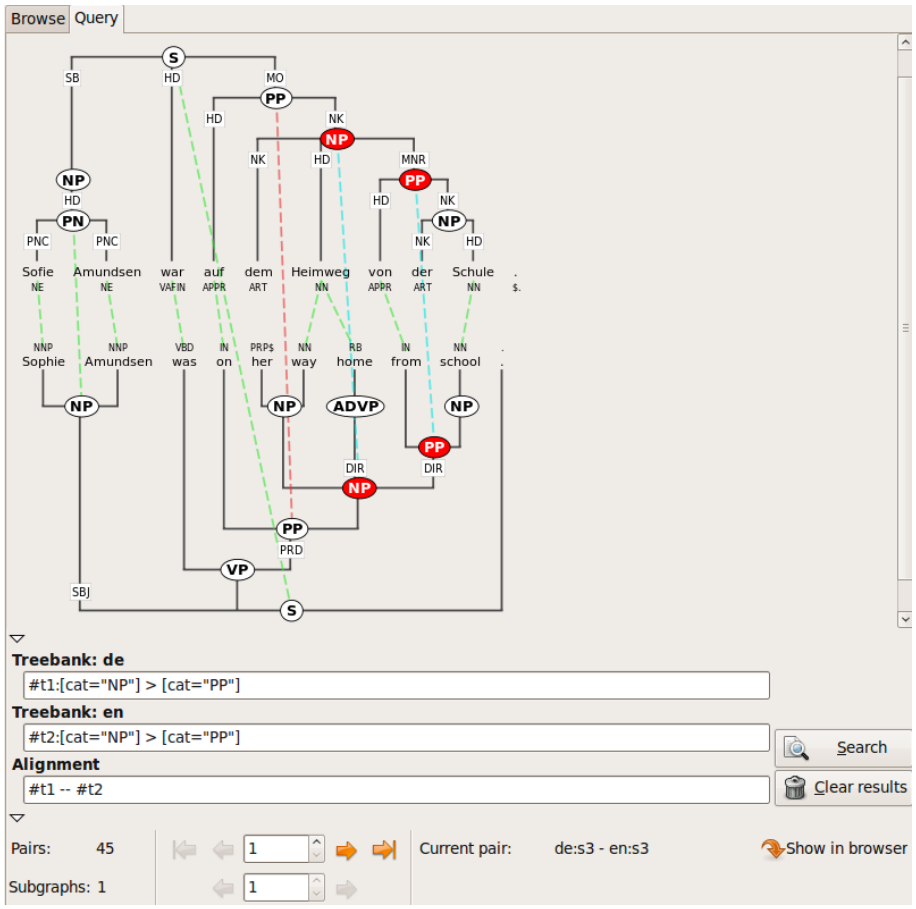


Abbildung 4.6: Suche im TreeAligner

Will man nur nach Alignierungen eines bestimmten Typs suchen, hängt man einfach an den Alignierungsoperator `--` den Typnamen an, also z.B. `--fuzzy`. Mit `?` in den Bedingungen für die Alignierung wird wahr zurückgegeben, wenn irgendeine Alignierung zwischen den beiden Graphen existiert.

Baumbank de:	`[NT] >RC [cat="S"]`
Baumbank en:	`[cat="SBAR"]`
Baumbank Alignierung:	`?`

Diese Anfrage gibt alle Baumpaare zurück, die Relativsätze enthalten und zwischen denen (irgendwelche) Knoten aligniert sind.
Indem nur in einem Baumbank-Feld eine Anfrage eingegeben wird, während die anderen beiden Felder leer bleiben, kann man die einzelnen Baumbanken separat durchsuchen. Als Resultat werden nur die Bäume in der gesuchten Sprache zurückgegeben.

Mit dem TreeAligner lassen sich die Alignierungen für neue parallele Baumbanken erstellen, andererseits ist er auch ein sehr mächtiges Werkzeug, um (nicht nur parallele) Baumbanken zu durchsuchen. Im nächsten Kapitel wird nun vorgestellt, wie der TreeAligner mit automatischen Vorschlägen für Wort- und Phrasenalignierungen erweitert wurde.

5 Automatische Alignierung im TreeAligner

Das Ziel dieser Arbeit war es, den im vorangehenden Kapitel vorgestellten TreeAligner mit automatisch generierten Vorschlägen für Wort- und Phrasenalignierungen zu erweitern. Die Vorschläge sollten pro Satz erfolgen und die manuelle Arbeit des Alignierens beschleunigen sowie die Konsistenz erhöhen. In den folgenden Unterkapiteln werde ich zunächst die Grundidee und die ersten Experimente mit Smultron erläutern, danach auf die verwendeten Algorithmen eingehen und im darauffolgenden Unterkapitel die Implementierung näher erklären. Im letzten Unterkapitel werden die Evaluierungsmethode und deren Ergebnisse dargelegt.

5.1 Experimente mit der Suche im TreeAligner

Als Ausgangslage für die automatischen Vorschläge sollen die bereits vorliegenden (manuellen oder manuell überprüften) Alignierungen des aktuellen Projekts (oder allenfalls diejenigen eines anderen Projekts) dienen. Mit jedem neu alignierten Satz sollen für künftige Vorschläge mehr Informationen hinzukommen. Als Testkorpora dienten die beiden Smultron-Teile, bei welchen die Alignierungen der letzten zehn Sätze gelöscht wurden. Da im TreeAligner eine Suchfunktion vorhanden ist, sollten erste Experimente zeigen, welche Resultate man bei der Wort- und Phrasensuche erhält und ob diese direkt als Vorschläge genutzt werden können. Nehmen wir das folgende Satzpaar als Beispiel:

(1) a. Sofie konnte ihrer Stimme anhören, daß es um etwas Ernstes ging.
b. Sophie could tell by her voice that it was something serious.

Beginnen wir beim ersten Wort im deutschen Satz; welches Wort im englischen Satz soll mit *Sofie* aus dem deutschen Satz aligniert werden? Aufschluss darüber gibt uns möglicherweise folgende Suchanfrage:

Baumbank de:	`#t1:[word="Sofie"]`
Baumbank en:	`#t2:[word=/.*/]`
Baumbank Alignierung:	`#t1 -- #t2`

Mit dieser Anfrage wird das Wort *Sofie* in der deutschen Baumbank gesucht, das mit irgendeinem Wort (ausgedrückt durch den regulären Ausdruck `.*`, alternativ könnte auch `#t2:[T]` verwendet werden) in der englischen Baumbank aligniert ist. Als Resultat werden 78 Satzpaare zurückgegeben. Da es sich in diesem einfachen Fall um einen Eigennamen handelt, ist der Wert für `#t2` immer derselbe, nämlich *Sophie*, ebenso wie der Alignierungstyp, welcher für dieses Knotenpaar durchgängig *good* ist. Also können wir nun in unserem englischen Satz (1-b) nach *Sophie* suchen, und zufälligerweise kommt es genau einmal vor. Das heisst, für das Satzpaar vom Beispiel (1) könnte vorgeschlagen werden, *Sofie* mit *Sophie* mit dem Alignierungstyp *good* zu alignieren. Beim nächsten Wort im Satz wird es bereits etwas kniffliger: Die Suche nach *konnte* (`#t1:[word="konnte"]`) gibt zwar nur 13 Satzpaare zurück und 11 davon sind *good*-Alignierungen mit *could*, einmal ist *konnte* jedoch mit *can* aligniert und in einem einzigen Fall mit *managed*. Im englischen Satz kommt glücklicherweise nur *could* vor, also sind *konnte* und *could* eindeutige Kandidaten für eine Alignierung. Die Suche nach *ihrer* ergibt gar keine Treffer. In anderen Fällen, vor allem bei Pronomen und Artikeln, kommt es häufig vor, dass es mehrere mögliche Kandidaten gibt, wie z.B. für das Pronomen *sie* in folgendem Beispiel:

(2) a. Hier konnte sie ganz sicher sein, daß niemand sie finden würde.
b. She knew - - nobody would find her there.

Hier gibt es für *sie* im englischen Satz aufgrund der Suchresultate drei mögliche Alignierungs-Kandidaten: *She*, "`--`" und *her*. In solchen Fällen muss also eine Strategie zur Disambiguierung angewandt werden, um den richtigen Kandidaten zu ermitteln. Wie eine solche Strategie aussehen könnte, wird in Kapitel 5.2 erläutert. Am häufigsten treten solche Ambiguitäten bei Artikeln, Pronomen, Präpositionen und Hilfsverben auf, da diese häufig mehrmals in einem Satz vorkommen.

5.1.1 1:n Alignierungen

Speziell zu beachten sind Mehrwortalignierungen. Da in den Smultron-Guidelines (vgl. Samuelsson et al. 2007: 3f) nur 1:n Wortalignierungen erwähnt werden, die Terminals also nicht m zu n aligniert werden sollten und die Phrasen nur eins zu eins, gehe ich nur auf den Fall von 1:n Wortalignierung ein. Um diese zu ermitteln,

kann folgende Suchanfrage gestellt werden:

```
Baumbank 1:              #t1:[T]
Baumbank 2:              #t2:[T] & #t3:[T] & #t2 .* #t3
Baumbank Alignierung:    #t1 -- #t2 & #t1 -- #t3
```

Die oberste Anfrage bedeutet, dass in der Baumbank 1 irgendein Terminal `#t1` gesucht ist. In der mittleren Zeile werden für die Baumbank 2 zwei beliebige Terminals `#t2` und `#t3` gesucht, wobei `#t3` (nicht unbedingt direkt) auf `#t2` folgt. Die Bedingung für die Alignierung auf der letzten Zeile ist nun, dass `#t1` sowohl mit `#t2` als auch mit `#t3` aligniert ist.
Wenn Baumbank 1 die englische Baumbank ist, liefert diese Anfrage 88 Baumpaare zurück, umgekehrt sind es 219. 1:n Alignierungen kommen also von Deutsch nach Englisch deutlich häufiger vor, was auf die Nominalkomposita im Deutschen zurückzuführen ist. So wird z.B. *Heimweg* mit *way* und *home* aligniert (siehe Abbildung 5.3). In der Richtung Englisch-Deutsch kommt das beispielsweise bei abgetrennten Verbzusätzen im Deutschen vor: So wird *imagine* mit *stell*, *dir* und *vor* aligniert. Diese Fälle müssen daher von den Mehrdeutigkeiten bei der Suchanfrage zum Pronomen *sie* im Beispiel (2) unterschieden werden: Sowohl die Suche nach *sie* als auch die Suche nach *Heimweg* liefern mehrere Resultate zurück; im ersten Fall ist aber nur eines der Resultate korrekt, im zweiten Fall sollten beide englischen Terminals mit dem deutschen aligniert werden.

5.1.2 Phrasenalignierungen

Um mögliche Phrasenalignierungen zu ermitteln, wird eine Suchanfrage etwas komplizierter. Die Bäume zu Beispiel (1) sind in Abbildung 5.1 graphisch dargestellt.
Ähnlich wie in Samuelsson und Volk (2007) (siehe Kapitel 3.3, S. 25) ist die Idee, die dominierten Wortfolgen einer Phrase mit denjenigen von bereits alignierten Phrasen zu vergleichen. Gesucht ist z.B. eine Nominalphrase, die das Wort *Sofie* (direkt oder indirekt) dominiert und mit irgendeinem Knoten aligniert ist. In der Suchanfrage würde das wie folgt aussehen:

```
Baumbank de:             #t1:[cat="NP"]>*[word="Sofie"]
Baumbank en:             #t2:[cat=/.*/]
Baumbank Alignierung:    #t1 -- #t2
```

Diese Anfrage gibt für `#t2` nur Nominalphrasenknoten zurück, die das Wort *Sophie* dominieren. Das ist ebenfalls der einfachste aller Fälle für Phrasen-Alignierung, da

sich im englischen Satz genau eine Nominalphrase findet, die das Wort *Sophie* dominiert, also könnte die deutsche Nominalphrase bedenkenlos mit der gefundenen englischen Nominalphrase aligniert werden. Im häufigeren Fall dominieren Phrasen nicht direkt Terminalknoten. Je höher oben im Baum die Phrase ist, desto komplexer wird die Suchanfrage.

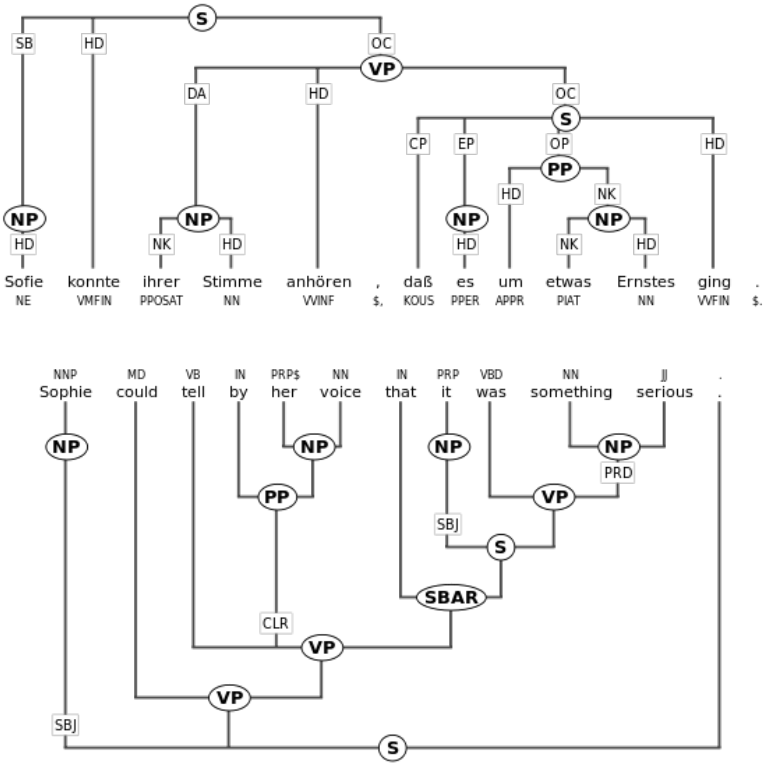


Abbildung 5.1: Beispiel (1) im TreeAligner

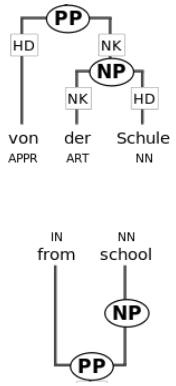


Abbildung 5.2: Zu alignierende Präpositionalphrasen

Für die Präpositionalphrase in Abbildung 5.2 müsste man die Anfrage so stellen:

Baumbank de:	`#t1:[cat="PP"] & #t3:[word="von"]`
	`& #t4:[word="der"]`
	`& #t5:[word="Schule"]`
	`& #t1 >* #t3 & #t1 >* #t4`
	`& #t1>* #t5 & #t3 . #t4 . #t5`
Baumbank en:	`#t2:[cat=/.*/]`
Baumbank Alignierung:	`#t1 -- #t2`

Der Ausdruck in der deutschen Spalte bedeutet zusammenfassend ausgedrückt: Gesucht ist ein Knoten mit der Kategorie PP, welcher mit einem Nonterminalknoten irgendeiner Kategorie aligniert ist und die Terminals *von*, *der* und *Schule* direkt oder indirekt dominiert, welche genau in dieser Reihenfolge vorhanden sind.
Diese Anfrage liefert zwei Resultate zurück. Einerseits genau die gesuchten beiden Präpositionalphrasen, andererseits auch die Präpositionalphrasen, die in Abbildung 5.3 rot markiert sind. Auch hier muss allenfalls noch disambiguiert werden.

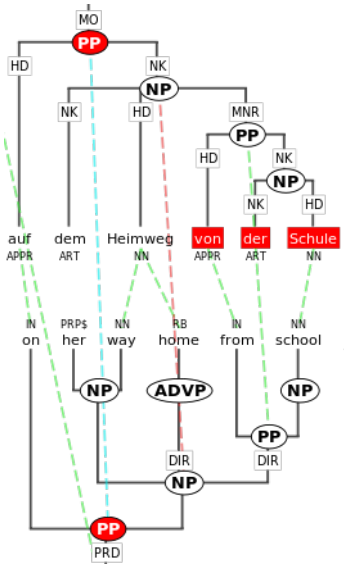


Abbildung 5.3: Resultat der Suchanfrage

Es kann bemängelt werden, dass nicht nach der ganzen Struktur gesucht wird – die Nominalphrase (siehe Abbildung 5.2) wird z.B. nicht in Betracht gezogen. Dies ist aber auch nicht nötig, da die Reihenfolge der dominierten Terminals eindeutig genug sein dürfte. Einzig in den Fällen, in denen ein Nonterminal genau einen Kindknoten dominiert, der ebenfalls ein Nonterminal-Knoten ist, sind die dominierten Terminals und deren Reihenfolge nicht mehr aussagekräftig genug (als Beispiel

siehe Abbildung 5.4).

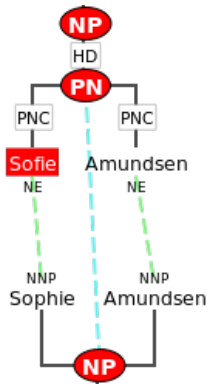


Abbildung 5.4: NP dominiert PN

Allerdings kommen diese Strukturen höchst selten vor, und zwar im deutschen und schwedischen Teil von Smultron, wie die Resultate der Anfrage

```
#nt:[cat!="VROOT"] & #nt2:[NT] & #t:[T] & #nt> #nt2 & #nt2 > #t
& arity(#t1,1)
```

[1]

zeigen (14 Vorkommen in der deutschen Baumbank). Dies ist darauf zurückzuführen, dass die ehemals flachen Strukturen dieser Baumbanken automatisch vertieft wurden, unter anderem mit folgenden Regeln (vgl. Samuelsson und Volk 2004: 3f):

- Wenn das direkte Kind einer Nominalphrase ein Adjektiv (POS=ADJA) ist, wird eine Adjektivphrase eingefügt.
- Wenn eine Nominalphrase ein Genitivattribut, eine Apposition oder einen pränominalen Genitiv (z.B. *Sofies Mutter*) direkt dominiert, wird eine Nominalphrase eingefügt.
- Wenn ein Satz-Knoten (S) oder eine koordinierte Nominalphrase (CNP) einen Mehrwort-Eigennamen (PN, ehemals MPN) direkt dominiert, wird eine Nominalphrase eingschoben (siehe unser Beispiel 5.4).
- Wenn ein Satz-Knoten (S), eine Verbalphrase (VP) oder eine koordinierte Nominalphrase einen nominalen Knoten (Nomen, Pronomen o.ä.) dominiert, wird eine Nominalphrase eingefügt.

Da die Regel für die Alignierungen aber lautet, immer den im Baum tiefer unten liegenden Knoten zu alignieren (wie auch in Abbildung 5.4 zu sehen ist), stellt diese Ambiguität in der Praxis kein Problem dar; die NP, die *Sofie Amundsen* (indirekt)

[1] Gesucht ist ein Knoten `#nt`, dessen Kategorie nicht `VROOT` ist, der genau einen (*arity*) beliebigen Nonterminal-Knoten `#nt2` direkt dominiert, welcher seinerseits direkt ein Terminal `#t` dominiert. Ausgeschlossen für das Nonterminal `#nt` wird die Kategorie `VROOT`.

dominiert, ist im Korpus nie aligniert und liefert so auf eine Anfrage auch keine Resultate zurück.
Natürlich gibt es bei der Suche von Phrasen viel weniger Resultate als bei der Wortsuche. Ambiguitäten treten dafür selten auf, hauptsächlich in den Fällen der Nominalphrasen, die ein Pronomen direkt dominieren, wie es im Beispiel (2) bei der Suche nach einer Alignierung der Nominalphrase, die das erste Personalpronomen *sie* dominiert, der Fall ist:

Baumbank de:	`#t1:[cat="NP"]>*[word="sie"]`
Baumbank en:	`#t2:[cat=/.*/]`
Baumbank Alignierung:	`#t1 -- #t2`

Der gesuchte Knoten im deutschen Baum und die Knoten im englischen Baum, die aufgrund der Suche als Kandidaten in Frage kommen, sind in Abbildung 5.5 rot markiert.

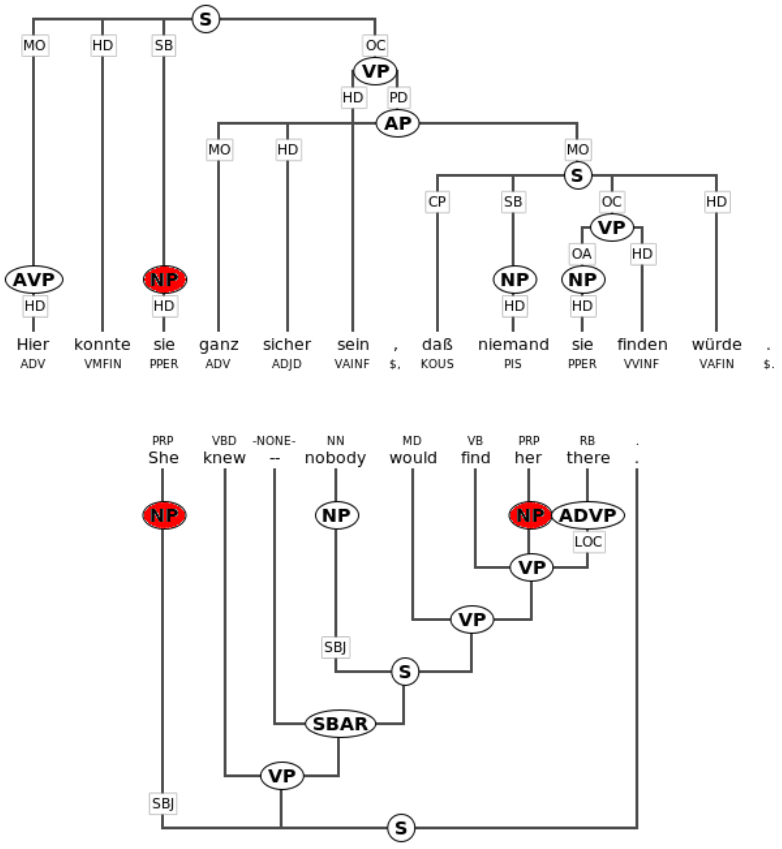


Abbildung 5.5: Mehrdeutige Nominalphrasen

5.2 Algorithmen

Wie wir im vorangehenden Kapitel festgestellt haben, liefert eine Suchanfrage bereits gute Anhaltspunkte über mögliche Alignierungskandidaten. Da die Resultate jedoch oftmals mehrdeutig sind, müssen sie für das aktuelle Satzpaar disambiguiert werden. Es werden die folgenden fünf Schritte durchlaufen, um zu den Vorschlägen zu gelangen:

1. Suche in den bereits vorhandenen Alignierungen nach möglichen Kanditaten im aktuellen, noch nicht alignierten Satzpaar
2. Disambiguierung der Wortalignierungsvorschläge
3. Disambiguierung der Phrasenalignierungsvorschläge
4. Auswahl der Kandidaten
5. Erweiterte Suche für Knoten ohne jegliche Suchresultate

Wie die Suche nach möglichen Kandidaten implementiert ist, werde ich in Kapitel 5.3 näher erläutern. In diesem Kapitel werden die Strategien zur Disambiguierung erklärt. Sie basieren auf einem Punktesystem: Für jedes Resultat der Suche werden zu Beginn Punkte vergeben. Die sicheren Alignierungen bekommen zunächst die höchste Punktzahl (10). *Sicher* heisst, dass sie schon vorhanden waren (d.h. sie wurden vom Annotator selbst erstellt oder zumindest für gut befunden), als die automatischen Vorschläge angefordert wurden. Wenn nur ein Vorschlag vorliegt, bekommt er eine höhere Startpunktzahl, als wenn für ein Wort oder eine Phrase mehrere Vorschläge in Frage kommen. Um die Strategien zu veranschaulichen, nehmen wir das Baumpaar in Abbildung 5.6 als Beispiel, mit den folgenden beiden Sätzen:

(3) a. Gleich darauf stand sie wieder im Wohnzimmer, und nun drückte sie Sofie in einen Sessel.
b. After a while she came back into the living room, and this time it was she who - - pushed Sophie into an armchair.

Für das gelb markierte Wort im deutschen Baum wurde ein Alignierungsvorschlag im englischen Baum gesucht. Die möglichen Kandidaten aufgrund einer Suchanfrage sind im englischen Baum grün markiert.

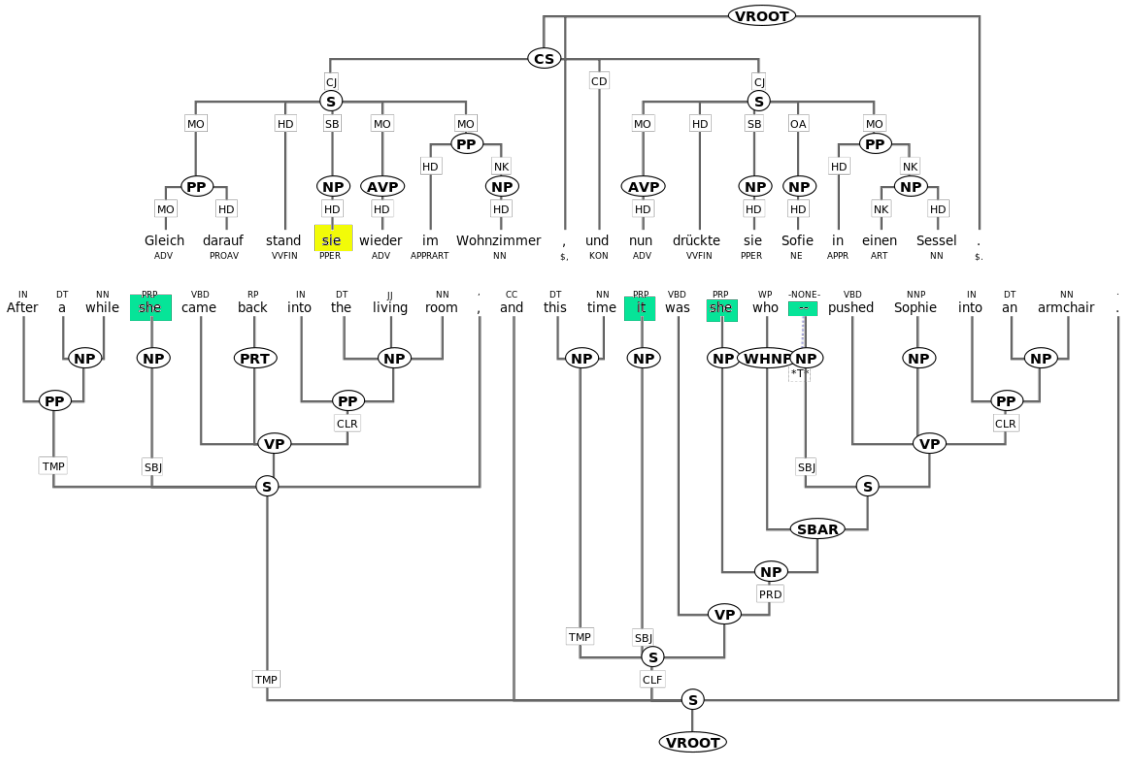


Abbildung 5.6: Sofies Welt, de:522 - en:521

In der Tabelle 5.1 sind noch weitere Alignierungskandidaten dieses Baumpaares aufgeführt, zur Vereinfachung allerdings nur ein paar ausgewählte. Die Punkte, die zu Beginn verteilt werden, sind ebenfalls in der Tabelle zu sehen. Für sichere Alignierungen gibt es 10 Punkte (das Beispiel enthält keine). Die sicheren Alignierungen werden besonders gekennzeichnet und sind von den Punktemodifikationen durch die Disambiguierungsstrategien nicht betroffen, d.h. sie sind Konstanten. Wenn nur ein Vorschlag für eine Phrase oder eine Mehrwortalignierung vorliegt, wie im Fall von *Wohnzimmer - living room* oder der Nominalphrasen, welche *Sofie* resp. *Sophie* dominieren, werden 9 Punkte vergeben. Bei den anderen Wortalignierungen wird bei nur einem Vorschlag als Startpunktzahl 8 vergeben, da sie etwas unsicherer sind als Phrasenalignierungen. Wenn es mehrere Vorschläge gibt, werden dafür je 6 Punkte vergeben, ausser wenn davon ein Wortpaar Teil einer Mehrwortalignierung ist, diese bekommen 7 Startpunkte (nicht im Beispiel). Mehrwortalignierungen werden bereits bei der Suche speziell markiert, wenn dasselbe Terminal im Quellsatz zu mehreren Terminals im Zielsatz Alignierungen aufweist. Sie werden auch für den weiteren Verlauf der Disambiguierung gekennzeichnet.

Dt.	**Engl.**	**Punkte**
sie_1	she_1	6
	it	6
	- -	6
	she $_2$	6
Wohnzimmer	living	9
	room	9
sie_2	she_1	6
	it	6
	- -	6
	she_2	6
Sofie	Sophie	8
NP (Sofie)	NP (Sophie)	9

Tabelle 5.1: Punktevergabe für Vorschläge von Wort- und Phrasenalignierungen des Baumpaares aus Abbildung 5.6

5.2.1 Disambiguierung von Wortalignierungsvorschlägen

Anhand unseres Beispiels des Pronomens *sie* aus dem Satzpaar (3) wollen wir nun die verschiedenen Strategien zur Disambiguierung der Wortalignierungsvorschläge betrachten. Die erste Strategie prüft, ob in der "anderen" Richtung Vorschläge für Mehrwortalignierungen gefunden werden (in den nachfolgenden Tabellen mit Str. 1 referenziert). Bis jetzt haben wir ja nur für die Wörter und Phrasen aus dem deutschen Satz Vorschläge für Alignierungen gesucht. Wenn es nun aber für ein englisches Wort Alignierungsvorschläge zu mehreren deutschen Wörtern gibt (z.B. von she_1 nach sie_1 und sie_2), wird nun zuerst überprüft, ob es sich dabei um eine Mehrwortalignierung handelt. Wenn das der Fall ist, bekommen die Vorschläge Pluspunkte, und sie werden als Mehrwortalignierung markiert. In unserem Beispiel tritt kein solcher Fall auf, jedoch würde z.B. *herself* mit *sich* und *selber* als Mehrwortalignierung markiert werden. Die Punktetabelle sieht nach diesem ersten Test also noch gleich aus wie in Tabelle 5.1.

Mit der zweiten Strategie wird geprüft, ob eindeutige Vorschläge für alignierte Elternknoten vorhanden sind, d.h. ob ein Alignierungsvorschlag für die Elternknoten vorliegt, der 8 oder mehr Punkte bekommen hat. Dies trifft auf den Vorschlag der Terminalknoten *Sofie* und *Sophie* zu, deren Elternknoten mit einer Punktzahl von 9 zur Alignierung vorgeschlagen sind. Die Elternknoten der Pronomen sind in der Tabelle zwar nicht aufgeführt, diese hätten jedoch ohnehin eine zu geringe Punktzahl, da sie ebenso mehrdeutig sind wie ihre Kindknoten.

Dt.	**Engl.**	**Str. 1**	**Str. 2**	**Punktestand**
sie_1	she_1	0	0	6
	it	0	0	6
	- -	0	0	6
	she_2	0	0	6
Wohnzimmer	living	0	0	9
	room	0	0	9
sie_2	she_1	0	0	6
	it	0	0	6
	- -	0	0	6
	she_2	0	0	6
Sofie	Sophie	0	+**3**	11
NP (Sofie)	NP (Sophie)	-	-	**9**

Tabelle 5.2: Punkte nach Überprüfung von Elternknoten-Alignierungen

Als Nächstes wird für die Vorschläge der Wortalignierungen der *mutual information score* berechnet. Die Berechnung der *mutual information* (MI) ist eine Vorgehensweise, die in der Computerlinguistik häufig bei der Suche nach Kollokationen eingesetzt wird. *Mutual information* $I(x, y)$ misst, wie gut eine zufällige Variable eine andere vorhersagt. Die Idee hinter dieser Methode geht auf Fano (1961)[2] zurück, der die wechselseitige Information zweier Ereignisse x und y wie folgt definierte:

$$I(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)}$$ [3]

$I(x, y)$ ist also die Menge der Information, welche die Zufallsvariable x über die Zufallsvariable y enthält, somit ein Korreliertheitsmass der gemeinsamen Information von x und y. Während I im Falle von selten vorkommenden, jedoch vollständig abhängigen x und y ansteigt, nimmt I im Falle vollständiger Unabhängigkeit den Wert 0 an. Im Falle von Kollokationen sind x und y zwei benachbarte Wörter eines Korpus, in unserem Fall sind x aus der Baumbank a und y aus der Baumbank b zwei alignierte Wörter einer parallelen Baumbank ab. Die Berechnungen der einzelnen Teile der obigen Formel für ein Wortpaar sind hier aufgeführt:

$$p(x) = \frac{Häufigkeit\ des\ Wortes\ x}{Alle\ Alignierungen}$$

[2] Robert Fano: Transmission of Information: A Statistical Theory of Communications. New York: Wiley and M.I.T. Press.

[3] Genau genommen ist diese Formel bereits eine abgewandelte Form der ursprünglichen und wird eigentlich *pointwise mutual information* genannt (vgl. Tiedemann 2003: 10).

$$p(y) = \frac{Häufigkeit\ des\ Wortes\ y}{Alle\ Alignierungen}$$

$$p(x,y) = \frac{Häufigkeit\ der\ Alignierung\ x,y}{Alle\ Alignierungen}$$

Die MI-Werte für unsere Beispielalignierungen sind in der folgenden Tabelle dargestellt.

Dt.	**Engl.**	**MI**	**Str. 1**	**Str. 2**	**Str. 3**	**Punkte**
sie_1	she_1	4.71	0	0	0	6
	it	1.77	0	0	**-1**	5
	- -	1.96	0	0	**-1**	5
	she_2	4.71	0	0	0	6
Wohnzimmer	living	11.20	0	0	**+1**	10
	room	10.20	0	0	**+1**	10
sie_2	she_1	4.71	0	0	0	6
	it	1.77	0	0	**-1**	5
	- -	1.96	0	0	**-1**	5
	she_2	4.71	0	0	0	6
Sofie	Sophie	5.86	0	+3	0	11
NP (Sofie)	NP (Sophie)	-	-	-	-	9

Tabelle 5.3: Punkte nach Einbezug der *mutual information*

Wenn der *mutual information score* kleiner als 3.5 ist, werden Minuspunkte vergeben, wenn er grösser als 8 ist, Pluspunkte. Diese Schwellwerte habe ich anhand von Stichproben festgelegt. Da die Werte aber je nach Korpusgrösse variieren können, wäre allenfalls eine automatische Methode zur Berechnung der Schwellwerte wünschenswert. Aus Tabelle 5.3 wird ersichtlich, dass nach diesem Schritt die Alignierungen zwischen $sie_{1/2}$ und *it* und $sie_{1/2}$ und -- die geringere Punktzahl aufweisen als die Alignierungen von $sie_{1/2}$ zu $she_{1/2}$. Da unsere Baumbank verhältnismässig klein ist für so ein Statistik-Mass, wird dieser Wert mit Vorsicht gehandhabt und mit +/- 1 Punkt nur leicht gewichtet im Vergleich zu den Punktvergaben der anderen Strategien. Auch kann eingewandt werden, dass die Häufigkeit, mit der eine Alignierung vorkommt, nicht relevant ist, da der menschliche Annotator die Alignierung ja beabsichtigt hat (ausser es handelt sich um einen Fehler). Wenn man sich allerdings zwischen zwei möglichen Alignierungen entscheiden muss, ist es sicher hilfreich, die häufigere zu bevorzugen.

Die letzte Vergabe von Punkten erfolgt mittels der Prüfung, ob in den Teilsätzen der Alignierungskandidaten sichere/eindeutige Alignierungsvorschläge vorliegen.

Dies dient zur Überprüfung, ob die beiden Kandidaten zu äquivalenten Teilsätzen gehören. Das sollte uns also z.B. ermöglichen, den Vorschlag der Alignierung von sie_1 zu she_2 auszuschliessen. Von den beiden vorgeschlagenen Terminals aus wird der nächste S-Knoten gesucht. Für dessen Kinder wird wiederum nach einem Alignierungsvorschlag mit 8 oder mehr Punkten gesucht. In denselben Teilbäumen wie sie_1 und she_1 befinden sich die Alignierungen von *Wohnzimmer* zu *living* und *room*, die beide beim momentanen Stand die Punktzahl 10 aufweisen, in denselben Teilbäumen wie sie_2 und she_2 wird die Alignierung zwischen *Sofie* und *Sophie* gefunden. Im idealen Setting unseres Beispiels würde das nun ausreichen, um diese Alignierungen derjenigen zwischen sie_1 und she_2 zu bevorzugen. Wenn keine Alignierung im selben Teilbaum gefunden wird, werden 2 Punkte abgezogen, falls eine gefunden wird, kommt ein Punkt hinzu. Ausgehend von den bisher in Betracht gezogenen Vorschlägen sähe unsere Punktetabelle demnach so aus:

Dt.	**Engl.**	**Str. 1**	**Str. 2**	**Str. 3**	**Str. 4**	**Punkte**
sie_1	she_1	0	0	0	**+1**	7
	it	0	0	-1	**-2**	3
	- -	0	0	-1	**-2**	3
	she_2	0	0	0	**-2**	4
Wohnzimmer	living	0	0	+1	**+1**	10
	room	0	0	+1	**+1**	10
sie_2	she_1	0	0	0	**-2**	4
	it	0	0	-1	**+1**	6
	- -	0	0	-1	**+1**	6
	she_2	0	0	0	**+1**	7
Sofie	Sophie	0	+3	0	**+1**	12
NP (Sofie)	NP (Sophie)	-	-	-	-	9

Tabelle 5.4: Punkte nach Strategie 4 unter idealer Annahme

Um zu vereinfachen, hatten wir bisher nur ein paar wenige Alignierungsvorschläge, die für das Beispielbaumpaar (siehe Beispiel (3) und Abbildung 5.6) gefunden werden, in unsere Berechnungen einbezogen. Auf die ersten drei Strategien hatte das bis jetzt keinen Einfluss, auf unsere letzte hingegen schon. Für das deutsche *stand* aus dem ersten deutschen Teilsatz (*Gleich darauf* ***stand*** *sie wieder im Wohnzimmer...*), wurde das englische Verb *was* aus dem zweiten englischen Teilsatz (*and this time it* ***was*** *she who...*) gefunden. Im ganzen Korpus wurde diese Alignierung nur einmal gefunden. Da dieser Vorschlag jedoch der einzige für *stand* (und auch der einzige für *was*) war, bekam er eine hohe Startpunktzahl (8) und da die Wortform *stand* im ganzen Korpus nur zweimal vorkommt, hilft hier auch der MI-Score nichts, d.h. die Alignierung wird als eindeutig gewertet und hat Einfluss auf das Resultat des

letzten Tests[4]. Unter Einbezug dieses Alignierungsvorschlags sieht die Punktetabelle nach dem Durchlauf der vierten Strategie tatsächlich so aus:

Dt.	**Engl.**	**Str. 1**	**Str. 2**	**Str. 3**	**Str. 4**	**Punkte**
sie_1	she_1	0	0	0	**+1**	7
	it	0	0	-1	**+1**	6
	- -	0	0	-1	**+1**	6
	she_2	0	0	0	**+1**	7
Wohnzimmer	living	0	0	+1	**+1**	10
	room	0	0	+1	**+1**	10
sie_2	she_1	0	0	0	**+1**	7
	it	0	0	-1	**+1**	6
	- -	0	0	-1	**+1**	6
	she_2	0	0	0	**+1**	7
Sofie	Sophie	0	+3	0	**+1**	12
NP (Sofie)	NP (Sophie)	-	-	-	-	9

Tabelle 5.5: Tatsächliche Punkte nach Strategie 4

Die Strategie hätte eigentlich unter anderem dazu dienen sollen, die Alignierung von sie_1 zu she_1 derjenigen von sie_1 zu she_2 zu bevorzugen: In der idealen Tabelle 5.4 sieht man, dass erstere sieben Punkte und letzere nur noch vier Punkte hat; die tatsächliche Punktzahl beträgt allerdings nun für beide Vorschläge sieben.
Die Punkteverteilung ist abgeschlossen, jetzt folgen noch die Entscheidungen. Als Grundsatz wird festgelegt, dass nur Knoten, deren Vorschläge als Mehrwortalignierung gekennzeichnet sind, 1:n aligniert werden dürfen, alle anderen Knoten müssen zwingend 1:1 aligniert werden. Mit Hilfe der Punkte erfolgt das erste Ausschlussverfahren: Nur die Vorschläge mit der höchsten Punktzahl werden behalten, alle anderen fallen weg. Für sie_1 bleiben dadurch she_1 und she_2 mit je sieben Punkten noch im Rennen, während *it* und -- mit je sechs Punkten ausgeschlossen werden. Als Nächstes wird der Alignierungstyp betrachtet: *good* wird *fuzzy* vorgezogen. Das hilft bei der Entscheidung zwischen sie_1 - she_1 (Vorschlag 1) und sie_1 she_2 (Vorschlag 2) auch nicht, der Alignierungstyp ist für beide Vorschläge *good*. Als letzte Massnahme zur Disambiguierung werden die Positionen der Terminals im Satz verglichen. Die Position von sie_1 ist 4 und auch she_1 ist das vierte Wort im Satz. An Position 17 steht she_2. Um die Distanz vom deutschen zum englischen Terminal zu ermitteln, wird der Absolutbetrag der Substraktion der Positionen berechnet. Die Distanz von Vorschlag 1 beträgt demnach 0, diejenige des Vorschlags 2 beträgt 13. Wenn die berechneten Distanzen nicht gleich sind und deren Differenz

[4]Zu testen bliebe, ob die MI-Werte eindeutiger wären, wenn sie über den Lemmata berechnet würden anstatt über Wortformen.

grösser als zwei ist, wird die kleinere Distanz bevorzugt, und der Vorschlag mit der grösseren Distanz wird ignoriert. Im Falle unserer zu alignierenden Pronomen bleibt jetzt nur noch Vorschlag 1 übrig, d.h. wir können sie_1 mit she_1 alignieren. Wenn nach all diesen Tests immer noch mehr als ein Kandidat übrig bleibt, wird kein Vorschlag zu diesen Kandidaten gemacht.

Hier noch eine Zusammenfassung der in diesem Unterkapitel beschriebenen Alignierungsstrategien, die in dieser Reihenfolge angewandt werden:

1. Ambige Vorschläge auf Mehrwortalignierungen prüfen
2. Prüfen, ob eindeutige Vorschläge für alignierte Elternknoten vorhanden sind
3. Mutual Information Score berücksichtigen: Bei sehr tiefem MI-Score Punktzahl senken, bei sehr hohem MI-Score Punktzahl erhöhen
4. Prüfen, ob Vorschläge für eindeutige Alignierungen im selben Satzteil vorhanden sind
5. Entscheidung:
   a) Nur mehrwort-gekennzeichnete Vorschläge dürfen (und müssen) 1:n aligniert werden, für alle andern gilt 1:1.
   b) Vorschläge mit höheren Punktzahlen werden bevorzugt.
   c) *good* vor *fuzzy*: Bei gleichem Punktestand und unterschiedlichem Alignierungstyp wird der Vorschlag, welcher mit *good* aligniert ist, bevorzugt.
   d) Relative Distanz vom Quell- zum Zielwort: Die Alignierung mit der kürzeren Distanz wird bevorzugt.

5.2.2 Disambiguierung von Phrasenalignierungen

Nachdem die Vorschläge zur Wortalignierung nun disambiguiert sind, können wir uns an die Disambiguierung der Phrasenalignierungen machen. Da vor allem die Knoten mit nur einem Kindknoten betroffen sind, war es notwendig, erst die Disambiguierung der Terminals abzuwarten, bevor man die Phrasenalignierungen disambiguiert. Schauen wir nochmal unser Beispiel (3) (siehe Abbildung 5.6) an. Für die NP, welche sie_1 dominiert (NP_{sie-1}), werden die NP-Knoten, welche she_1 (NP_{she-1}) und she_2 (NP_{she-2}) dominieren, zur Alignierung vorgeschlagen. Also wählen wir denjenigen Knoten des Zielsatzes, dessen Kindknoten mit dem/den Kindknoten von NP_{sie-1} aligniert ist/sind. Das trifft nach der Disambiguierung der Terminals auf die (NP_{she-1} zu. Dies ist die einzige Disambiguierungsstrategie, die bei den Phrasenalignierungen

angewandt wird. Wenn dadurch die Ambiguität nicht aufgelöst werden kann, wird für diese Knoten kein Vorschlag gemacht.

5.2.3 Suche nach weiteren Kandidaten

Dieselbe Strategie, die zur Disambiguierung von Phrasenalignierungen verwendet wurde, können wir auch benutzen, um weitere Phrasenalignierungen zu finden. Nehmen wir an, dass in unserem Baumpaar NP*[die schwarze Katze]* und NP*[the black cat]* vorkommt, im Korpus aber nur Alignierungen zwischen *die Katze* mit *the cat* vorhanden sind (sowohl zwischen den einzelnen Terminals als auch zwischen den beiden NPs). Im Korpus sind aber auch auch die beiden NPs NP*[der schwarze Hund]* und NP*[the black dog]* vorhanden, deren Adjektive *schwarz* und *black* miteinander aligniert sind. Alle Kindknoten der deutschen NP*[die schwarze Katze]* sind also mit den Kindknoten der englischen NP*[the black cat]* aligniert. Ist dies der Fall, kann man annehmen, dass auch deren Elternknoten aligniert werden. Diese Regel kommt dann zur Anwendung, wenn die Alignierungen der Kindknoten auch alle vom Typ *good* sind. Bei einer oder mehreren *fuzzy*-Alignierungen wird es etwas schwieriger: Können die Elternknoten wirklich aligniert werden, wenn die Kindknoten teilweise ungenaue Übersetzungen von einander sind? Aligniert man sie in dem Fall auch *fuzzy*? Oder vielleicht doch besser gar nicht und überlässt diese Entscheidung dem menschlichen Annotator?

Weitere Terminal-Alignierungen könnten z.B. über das Lemma gefunden werden. Wir erinnern uns, dass für das Possessivpronomen *ihrer* aus Beispiel (1) im englischen Satz kein Äquivalent gefunden wurde, da es im Korpus bisher einfach nie aligniert war. Ziehen wir nun aber anstelle des Wortes das Lemma und das POS-Tag des Terminals in Betracht, finden wir unter anderem in der parallelen Baumbank die Alignierung von *ihre [Mutter]* zu *her [mother]* und stellen fest, dass *her* mit demselben POS-Tag im englischen Satz (1-b) ebenfalls vorhanden ist und bisher noch nicht aligniert wurde. Leider funktioniert dieses Vorgehen nur für deutsche und schwedische Terminals; in der englischen Baumbank sind die Lemmata nicht vorhanden.
Durch Kompositazerlegung im Deutschen lassen sich allenfalls auch noch weitere Alignierungen finden. Wenn z.B. *Garten* mit *garden* und *Tor* mit *gate* aligniert wurden, *Gartentor* aber bisher noch nie vorgekommen ist, kann das deutsche Wort mit Hilfe der Kompositazerlegung trotzdem mit dem englischen *garden gate* aligniert werden.
Für Eigennamen, welche in Textsorten wie Zeitungstexten o.ä. nur selten bereits

bekannt sind (d.h. bereits im Korpus schon einmal erwähnt und auch aligniert wurden), könnten Ähnlichkeitsmasse wie die Levenshtein-Distanz[5] bei der Entscheidung helfen, ob sie *good* (bei sehr wenig Abweichung wie im Fall von *Sofie* und *Sophie*), *fuzzy* (bei höherer Abweichung wie im Fall von *Jorunn* und *Joanna* oder, bei zu hoher Abweichung, gar nicht aligniert werden sollten.

5.3 Umsetzung

In den folgenden Unterkapiteln werde ich zunächst die Struktur des TreeAligner-Codes, bzw. diejenige der Bibliothek *Ladon* näher beschreiben, danach auf den Aufbau der verwendeten Datenbanken eingehen und zum Schluss noch die Einbettung der automatischen Alignierungs-Vorschläge in Ladon darlegen.

5.3.1 Ladon: Die Bibliothek zum TreeAligner

Seit der Version 1.1.90 des TreeAligners[6] wurden seine Hauptfunktionen in eine Bibliothek namens *Ladon*[7] ausgelagert. Im TreeAligner-Projekt selbst sind hauptsächlich noch die Klassen zur Darstellung der Benutzeroberfläche enthalten, der Rest wird von Ladon importiert. Ladon enthält den Code zur Verarbeitung und zum Durchsuchen der Tigerkorpora, für die Alignierungsprojekte und -vorschläge sowie für die FrameNet-Erweiterung und das Zeichnen der Bäume. Die Paketstruktur in Ladon sieht folgendermassen aus (vgl. Marek 2009: 65ff):

- *Alignierung*: `ladon.align`
  Handhabung von parallelen Korpora mit syntaktischen Alignierungen
- *Graphenrendering*: `ladon.drawing`
  System-unabhängiges Rendering der Syntaxgraphen
- *Frame-Datenbank*: `ladon.framenet`
  Laden von FrameNet-Dateien

[5] Die Levenshtein-Distanz (auch Edit-Distanz, Editierdistanz oder Editierabstand) bezeichnet in der Informationstheorie ein Mass für den Unterschied zwischen zwei Zeichenketten bezüglich der minimalen Anzahl der Operationen Einfügen, Löschen und Ersetzen, um die eine Zeichenkette in die andere zu überführen. Die Levenshtein-Distanz zwischen *Tier* und *Tor* beträgt beispielsweise 2:
1. Ersetze *i* durch *o*
2. Lösche *e*

[6] Quellcode: `http://www.cl.uzh.ch/kitt/hg/sta/master/`

[7] Quellcode: `http://www.cl.uzh.ch/kitt/hg/ladon/master/`

- *Tiger*: `ladon.tiger`
  Parser für TigerKorpora, Korpusindex-Erstellung und -Zugang, Abfragesprache
- *Hilfsdateien*: `ladon.utils`
  Diverse Hilfsfunktionen und -klassen

In das Paket `ladon.align` werden die automatischen Vorschläge eingebettet.

5.3.2 Datenbanken

Für die automatischen Vorschläge werden zwei zusätzliche Sqlite-Datenbanken[8] erstellt, die Alignierungs-Datenbank und die Disambiguierungs-Datenbank. Erstere wird generiert, wenn für ein Projekt das erste Mal Alignierungsvorschläge angefordert werden. Die zweite Datenbank ist eine temporäre, die für jede Suchanfrage neu aufgebaut wird und dazu dient, die Bepunktung der Vorschläge zu aktualisieren. Daraus werden letztlich die Vorschläge extrahiert. In den folgenden beiden Unterkapiteln werde ich den Aufbau der Datenbanken näher beschreiben.

5.3.2.1 Alignierungs-Datenbank

Die Klassen und Funktionen zur Erstellung und Abfrage dieser Datenbank sind im Modul `ladon.algin.autoalign.align_db` zu finden. Die Datenbank besteht aus drei Tabellen:

Terminal-Tabelle: In dieser Tabelle werden die alignierten Terminals mit IDs und Feature-Werten gespeichert.
Bsp.: `15|IndexNodeId(2, 11)|0|heimweg|Heim#weg|NN|MASK|5`
Die Felder der Tabelle in der Reihenfolge des Beispiels:

- ID: Primärschlüssel, mit dem die Tabelle indexiert wird
- Knoten-ID: In der ID sind die Satz- und Knotennummer des Tigerknotens enthalten, im Beispiel `IndexNodeId(2, 11)`
- Baumbank-ID: `0` steht in diesem Fall für die deutsche Baumbank
- Wort: Wort des Terminalknotens in Kleinbuchstaben, im Beispiel `heimweg`
- Lemma: Lemma-Angabe, falls vorhanden, im Beispiel `Heim#weg`
- POS: Part-of-Speech des Terminals, im Beispiel `NN`

[8] `http://www.sqlite.org/`

- Morph: Morphologie-Angaben, falls vorhanden, im Beispiel `MASK`
- Ordnung: Die Position des Terminals im Satz, im Beispiel `5`

Nonterminal-Tabelle: In dieser Tabelle werden die alignierten Nonterminals mit IDs und dem Feature `cat`, sowie mit den dominierten Terminals gespeichert.
Bsp.: `1|IndexNodeId(2, 15)|PP|1|from school`
Die Felder der Tabelle in der Reihenfolge des Beispiels:

- ID: Primärschlüssel, mit dem die Tabelle indexiert wird
- Knoten-ID: In der ID sind die Satz- und Knotennummer des Tigerknotens enthalten, im Beispiel `IndexNodeId(2, 15)`
- Kategorie: Kategorie des Knotens, im Beispiel `PP`
- Baumbank-ID: `1` steht für die englische Baumbank
- Wörter: Zeichenkette der Terminals, welche vom Knoten dominiert werden, im Beispiel `from school`

Alignierungs-Tabelle: Hier werden die Alignierungen zwischen den Knoten und der Alignierungstyp gespeichert. Die Reihenfolge der Felder entspricht den IDs der Baumbanken, d.h. ins erste Feld werden alle Knoten-IDs der deutschen Baumbank gefüllt, ins zweite diejenigen der englischen.
Ein Eintrag: `IndexNodeId(2, 11)|IndexNodeId(2, 14)|good`
Die Felder in der Reihenfolge des Beispiels:

- 1. Knoten-ID: `IndexNodeId(2, 11)`
- 2. Knoten-ID: `IndexNodeId(2, 14)`
- Alignierungstyp: `good`

Die Datenbank wird mit den Daten der bereits vorhandenen Alignierungen gefüllt. Diese Daten sind zwar bereits in anderer Form in einer Datenbank vorhanden, werden aber in der Alignierungs-Datenbank in einer optimierten Form gespeichert, um eine effiziente Suche zu ermöglichen. Um also vorhandene Alignierungen eines bestimmten Wortes oder einer bestimmten Phrase zu finden, werden Anfragen an die Datenbank gestellt. Die Resultate der Anfragen werden danach weiter verarbeitet (disambiguiert etc.). Neue Alignierungen werden der Datenbank hinzugefügt, und gelöschte Alignierungen werden entfernt. Über die Funktion `search_terminal_alignments` wird folgende Suchanfrage an die Datenbank gestellt, wenn wir beispielsweise wissen wollen, ob und womit das Wort *Heimweg* im Korpus aligniert ist:

```
SELECT t1.id, t2.*, a.alignment_type FROM terminals t1
 INNER JOIN alignments a ON (t1.node_id=a.node_id_0)
```

```
INNER JOIN terminals t2 ON (a.node_id_1=t2.node_id and
                            t2.treebank_id=1)
WHERE t1.word = "heimweg" AND t1.treebank_id = 0
```

Die erste Zeile bedeutet: "Selektiere das Feld id von der Tabelle `terminals`, welche mit dem Alias `t1` abgekürzt wird, sowie alle Felder der Tabelle `terminals`, deren Auswahl an den Alias `t2` geknüpft ist und den Alignierungstyp des Alias `a`." Die sogenannten *inner joins* der zweiten und dritten Zeile sind nötig, um die Anfrage über mehrere Tabellen zu verknüpfen.
Die zweite Zeile verknüpft die Alignierungs-Tabelle `alignments`, abgekürzt mit `a`, mit `t1`: Die Knoten-ID von `t1` muss gleich sein der Knoten-ID `node_id_0` von `a`. In der dritten Zeile wird die Tabelle `terminals` über den Alias `t2` mit sich selbst verknüpft. Die Knoten-ID von `t2` muss gleich dem Feld `node_id_1` von `a` sein, und die Treebank-ID soll 1 entsprechen. Die letzte Zeile, das Where-Statement, definiert näher, wie `t1` auszusehen hat: Der Wert des Feldes `word` soll "heimweg" sein, und die Baumbank-ID von `t1` soll den Wert 0 haben.

5.3.2.2 Disambiguierungs-Datenbank

Diese Datenbank wird im Modul `ladon.algin.autoalign.disambiguator` erstellt. Wie oben erwähnt, wird sie nicht dauerhaft gespeichert; sie ist nur während der Disambiguierungsphase vorhanden. Die Datenbank enthält die Tabelle `proposals`, welche folgende Felder beinhaltet:

- ID `[id]`: Primärschlüssel
- Knoten-ID 1 und 2 `[node_id_0], [node_id_1]`: Diese beiden IDs sind *unique*, was sicherstellt, dass es keine doppelten Einträge gibt. Die Reihenfolge ist relevant: Die erste ID entspricht dem Knoten aus der ersten Baumbank, die zweite demjenigen aus der zweiten Baumbank. Welche Baumbank die "erste" ist, wird ganz zu Beginn festgelegt und entspricht der Reihenfolge, in der die Baumbanken ins Projekt eingefügt wurden.
- Punkte `[score]`: In diesem Feld wird die aktuelle Punktzahl des Alignierungsvorschlags gespeichert.
- Mutual Information Score `[mi]`: In diesem Feld wird der *Mutual Information Score* (siehe Kapitel 5.2.1, S.48) gespeichert, allerdings nur für Terminalalignierungen.
- Mehrwortalignierung `[multi]`: Dieses Feld ist eine Boolesche Variable, die anzeigt, ob diese Alignierung Teil einer Mehrwortalignierung ist oder nicht.

- Automatisch `[auto]`: Diese Boolesche Variable wird auf den Wert *false* gesetzt, wenn die Alignierung bereits vorhanden war, als die Vorschläge angefordert wurden.
- Abgelehnt `[deprecated]`: Dieser Wert wird ganz am Schluss gesetzt. Diese Boolesche Variable ist *true*, wenn die Alignierung nicht mehr in Betracht gezogen wird, und wird auf *false* gesetzt, wenn die Alignierung als eindeutig erkannt wurde.
- Typ `[type]`: Der Alignierungstyp wird hier gespeichert, d.h. der Wert des Feldes ist entweder *good* oder *fuzzy*.
- Terminal/Nonterminal `[tnt]`: Ist der Wert des Feldes "T", handelt es sich um eine Alignierung zwischen Terminals. Wenn der Wert "NT" ist, handelt es sich um eine Nonterminal-Alignierung. Terminal-zu-Nonterminal-Alignierungen werden nicht berücksichtigt.

Die Disambiguierungs-Datenbank ist im Gegensatz zur Alignierungs-Datenbank nicht in erster Linie zur Abfrage da, sondern dient der Datenmodifikation. Die Strategien zur Disambiguierung beeinflussen die verschiedenen Werte der Felder.

5.3.3 Einbettung in Ladon

Die Erweiterung der automatischen Alignierungsvorschläge ist, wie der TreeAligner bzw. Ladon auch, in Python programmiert. Im Paket `ladon.align.auto_align` sind die Klassen und Funktionen abgelegt, welche die automatischen Alignierungen berechnen. Die in Kapitel 5.3.2.1 beschriebene Alignierungs-Datenbank wird über das Projekt (die Klasse `ladon.align.project.AlignmentProject`) initialisiert und aufgerufen. Die Klasse `AlignmentGenerator` ist die Hauptklasse der automatischen Alignierung; von ihrer Methode `align_clear_candidates` werden alle anderen Klassen aufgerufen, über welche die automatischen Vorschläge berechnet werden. Als Erstes werden die Terminal- und die Nonterminalvorschläge über die Klassen `TerminalAlignments` und `NonTerminalAlignments` erstellt. Deren Methode `db_request` iteriert durch den aktuellen Graphen, für den die automatische Alignierung angefordert wurde, und sucht für jedes Terminal bzw. Nonterminal Alignierungen in der Datenbank (ähnlich dem Vorgehen in Kapitel 5.1). Um die Resultate kompakt darzustellen, sind die beiden Klassen `TerminalResults` und `NonTerminalResults` zuständig. Dort wird z.B. der wahrscheinlichste Typ berechnet und die Häufigkeit einer Alignierung zurückgegeben.
Mit diesen Resultaten ruft `align_clear_candidates` als Nächstes den `Disambiguator` auf. Wie der Name schon sagt, ist das die Klasse, welche für

die Disambiguierung der Suchresultate zuständig ist. Als Erstes baut sie aus den übergebenen Resultaten die Disambiguierungsdatenbank auf (siehe Kapitel 5.3.2.2). Danach werden zuerst die verschiedenen Methoden zur Disambiguierung der Wortvorschläge aufgerufen. Zum Schluss werden noch die Nonterminal-Alignierungen disambiguiert.

Die Methoden werden in der in Kapitel 5.2.1 beschriebenen Reihenfolge abgearbeitet. Jede Methode aktualisiert die Datenbank. Zuerst wird in `multiple_alignments_test` nach weiteren möglichen Mehrwortalignierungen gesucht. Bei einem Treffer werden die Punkte dieser Alignierungsvorschläge in der Datenbank erhöht und das Feld `multi` für die betroffenen Einträge auf den Wert *true* gesetzt. Als Nächstes wird mit der Methode `aligned_parents_test` festgestellt, ob ein Alignierungsvorschlag alignierte Elternknoten besitzt. Falls ja, wird in der Datenbank die Punktzahl der betreffenden Einträge erhöht. Der `frequency_test` hat einen etwas irreführenden Namen. Tatsächlich wird hier je nach Höhe des MI-Scores eines Alignierungsvorschlags die Punktzahl erhöht, gesenkt oder gleich belassen. Die Schwellwerte dafür habe ich anhand von Stichproben festgelegt und müssten eventuell je nach Korpusgrösse auch noch angepasst werden. Momentan werden Punkte abgezogen, wenn der MI-Score unter 3.5 liegt, und Punkte hinzugefügt, wenn er einen Wert von 8 überschreitet. Die letzte Methode, in der Punkte verteilt werden, ist `alignments_in_subtree_test`. Diese Methode prüft, wie bereits in Kapitel 5.2.1 beschrieben, ob sichere Alignierungen in denselben Teilbäumen der jeweiligen Knoten vorhanden sind. Wenn etwas gefunden wird, wird der Punktestand dieses Vorschlags in der Datenbank um einen Punkt erhöht, andernfalls um zwei Punkte gesenkt.

Ebenfalls wie in Kapitel 5.2.1 werden nun die ambigen Vorschläge miteinander verglichen und zuerst nach Punktestand, dann nach Typ und zum Schluss noch nach Position im Satz ausgefiltert. Wenn sich beim Vergleichen ein oder mehrere Vorschläge als schlechter erweisen, wird für diese in der Datenbank der Wert `deprecated` *true* gesetzt. Bleibt nach einem Vergleich nur noch ein Vorschlag für ein Knotenpaar übrig, so wird bei dessen Eintrag `deprecated` auf den Wert *false* gesetzt. Wenn alle Vergleiche durchlaufen wurden und für ein Knotenpaar noch immer mehr als ein Vorschlag vorhanden ist, werden bei beiden die Werte von `deprecated true` gesetzt; denn wenn wir uns nicht entscheiden können, machen wir lieber keinen Vorschlag als einen falschen.

Wie die die Datenbankeinträge nach Abschluss aller Tests aussehen, ist hier dargestellt. Die Felder von links nach rechts sind `id`, `node_id_0`, `node_id_1`, `score`, `mi`, `multi`, `auto`, `deprecated`, `type`, `tnt`:

```
sqlite> select * from proposals;
1|523,4|522,5|8|8.14762427646711|0|1|0|good|T
2|523,3|522,3|12|5.85613729514388|0|1|0|good|T
3|523,12|522,14|9|5.65260390105875|0|1|0|good|T
4|523,18|522,21|9|7.64344627499973|0|1|0|good|T
5|523,14|522,17|12|5.73301532168596|0|1|0|good|T
6|523,2|522,2|9||0|1|0|good|NT
7|523,13|522,16|9||0|1|0|good|NT
```

Die Booleschen Werte werden mit 0 für *false* und 1 für *true* ausgedrückt. Als nächster Schritt wäre nun der `reconstructor` an der Reihe. Diese Klasse ist dafür gedacht, weitere Alignierungsvorschläge für Knoten zu finden, die bis zu diesem Punkt "vorschlagslos" geblieben sind. Dies ist bis jetzt aber noch nicht ausprogrammiert. Als letzter Schritt generiert der `AlignmentGenerator` effektiv die Vorschläge, indem alle Einträge der Datenbank extrahiert werden, deren Wert von `deprecated` auf *false* gesetzt ist.

Eine Übersicht über die relevanten Klassen und Methoden ist in Abbildung 5.7 als Klassendiagramm zu sehen.

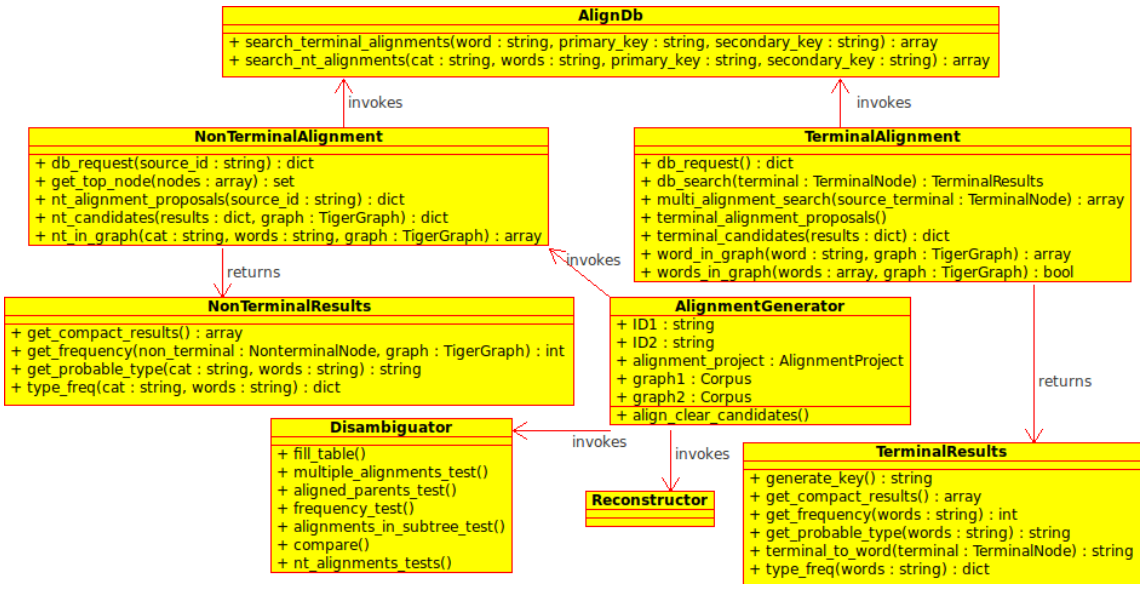


Abbildung 5.7: Klassendiagramm für die `auto_align`-Klassen

5.4 Evaluation

5.4.1 Wie/was evaluieren?

Mit welchen Methoden können die automatischen Vorschläge im TreeAligner aligniert werden? Einerseits über Masse wie Präzision (‘*precision*’) und Ausbeute (‘*recall*’), andererseits auch über qualitative Analysierung von Stichproben. Um zu sehen, wie und ob sich die Präzision und die Ausbeute mit mehr Datenmengen als Grundlage verbessern, habe ich ein kleines Python-Skript geschrieben (siehe Anhang B, S.81). Diesem übergibt man eine fertig alignierte parallele Baumbank (in unserem Fall die beiden Teile von Smultron), deren Alignierungen als Goldstandard dienen. In einer Kopie des Projekts wird durch die Hälfte aller alignierten Baumpaare iteriert. In jedem Durchgang werden die Alignierungen des aktuellen Baumpaares gelöscht und die automatischen Alignierungen generiert, beginnend mit dem letzten Baumpaar der Baumbank. Dadurch wird die Datengrundlage simuliert, die auch ein menschlicher Annotator zur Verfügung hat, wenn er am Projekt arbeitet; wenn erst die Hälfte des Projektes aligniert ist, stehen nur diese Informationen zur Verfügung, und mit weiterem Fortschreiten des Alignierens wird die Datenbasis grösser und hat auch Einfluss auf die Alignierung. Das Skript berechnet für jedes Baumpaar die Ausbeute und die Präzision der Terminals und Nonterminals einzeln, die Präzision des Typs sowie das F-Mass. Die Ausbeute zeigt, wie viele korrekte Alignierungen extrahiert wurden. Die Präzision hingegen ist ein Mass dafür, wie viele der gefundenen Alignierungen auch korrekt sind. Das F-Mass schliesslich kombiniert Präzision und Ausbeute. Die Formeln für diese drei Masse:

$$Ausbeute : \frac{gefundene\ korrekte\ Alignierungen}{korrekte\ Alignierungen}$$

$$Präzision : \frac{gefundene\ korrekte\ Alignierungen}{gefundene\ Alignierungen}$$

$$F - Mass : \frac{1 + \beta^2}{\frac{\beta^2}{Ausbeute} + \frac{1}{Präzision}}$$

Beim F-Mass wurde für β 1 angenommen. Je nachdem, wie Ausbeute und Präzision gewichtet werden, kann dieser Wert angepasst werden.

5.4.2 Ergebnisse

Ich habe das Evaluations-Skript sowohl über den Sofie-Teil als auch über den Wirtschaftsteil von Smultron laufen lassen. Das Erste, was dabei auffällt, ist, dass es für den Sofie-Teil 3.6 Minuten braucht, um ca. 264 Satzpaare zu alignieren, während die Dauer für die automatische Alignierung von 259 Satzpaaren des Wirtschaftsteils ganze 22 Minuten beträgt. Das liegt ganz klar an der Satzlänge und der komplexeren Struktur der Wirtschaftstexte. Zum Vergleich sieht man in Tabelle 5.6 die drei Teile der Baumbanken (inkulsive dem neuen Handbuch-Teil, dem noch Spanisch als Sprache hinzugefügt wurde, vgl. Göhring (2009)). Für die Berechnung der Satzlänge wurden die Tokens gezählt, d.h. auch Interpunktion o.ä. ist darin enthalten.

	Anzahl Sätze				Durchschnitt Tokens pro Satz			
	EN	DE	SV	ES	EN	DE	SV	ES
Sofies Welt	528	529	536		14.83	14.02	13.79	
Wirtschaftstexte	473	510	492		24.58	21.28	19.20	
DVD-Handbuch	512	547	533	518	20.08	16.43	16.66	20.92

Tabelle 5.6: Zahlen zu Smultron: Anzahl Sätze und durchschnittliche Satzlängen

Im Vergleich zu den anderen beiden Teilen sind die Sätze des Sofie-Teils also viel kürzer, was auch auf einfachere Strukturen schliessen lässt. Dies begünstigt die automatische Alignierung, wie aus den Tabellen 5.7 und 5.8 herauszulesen ist. Für die Werte der Tabellen, wurde über die Ergebnisse iteriert und nach jedem 20. Satzpaar der Durschnitt genommen. Das berechnete F-Mass setzt sich sowohl aus den Werten für die Nonterminals als auch aus denjenigen für die Terminals zusammen. Ausb. ist die Abkürzung für Ausbeute, Präz. steht für Präzision. Tabelle 5.7 sind die Durchschnittswerte für den Sofie-Teil, Tabelle 5.8 beinhaltet die Werte für den Wirtschaftsteil.

Start de:en	**Ende de:en**	**T-Ausb.**	**T-Präz.**	**NT-Ausb.**	**NT-Präz.**	**F-Mass**
264:257	290:283	36.54	80.34	14.86	56.35	40.58
291:284	310:302	40.30	78.67	22.99	62.50	44.15
311:303	330:320	32.25	74.80	12.35	33.33	36.61
331:321	348:341	31.03	59.17	22.89	56.67	37.40
349:342	368:360	52.55	76.62	30.10	74.83	54.41
369:361	388:380	44.53	87.74	29.02	65.00	50.37
389:381	407:400	35.32	80.00	21.32	75.00	42.90
408:401	427:423	38.48	85.34	22.07	65.00	44.50
428:424	447:443	39.97	80.26	24.59	75.00	46.54

Start de:en	Ende de:en	T-Ausb.	T-Präz.	NT-Ausb.	NT-Präz.	F-Mass
448:444	466:464	42.24	78.20	21.96	60.83	45.41
467:465	485:483	34.89	71.36	17.38	56.67	40.27
486:484	508:507	48.44	80.44	35.51	87.50	54.19
509:508	528:527	36.10	74.25	33.02	77.50	45.55

Tabelle 5.7: Durchschnittliches F-Mass der automatischen Vorschläge für den Sofie-Teil von Smultron

Weiter zusammenfassend beträgt somit die Ausbeute der Terminals zwischen 31% und 48%, und deren Präzision liegt zwischen und 71% und 88%. Bei den Nonterminals liegt die Ausbeute einiges tiefer, zwischen 15% und 33%, und auch die Werte der Präzision sind etwas geringer, nämlich im Schnitt zwischen 33% und 87%. Die Werte für das F-Mass sind alle relativ ähnlich zwischen 40% und 50%, nur bei ein paar wenigen Abschnitten schlägt es nach unten (tiefster Durchschnittswert: 36.61) und oben (höchster Durchschnittswert: 54.41) aus. Der Durchschnittswert für die Präzision des Alignierungstyps liegt bei 89.35% relativ hoch.

Start de:en	Ende de:en	T-Ausb.	T-Präz.	NT-Ausb.	NT-Präz.	F-Mass
258:227	296:261	30.57	59.64	9.28	50.43	33.18
320:286	297:262	21.33	51.54	2.36	15.00	22.67
321:287	339:306	37.63	48.84	7.88	20.25	34.22
340:307	358:326	26.72	55.59	6.41	25.00	28.02
359:327	378:346	34.68	57.83	9.85	17.92	31.52
379:347	398:362	45.62	44.58	17.17	55.83	37.86
399:363	419:382	52.97	52.71	20.22	57.92	45.16
420:383	439:399	53.30	51.25	21.85	60.00	45.86
440:399	458:414	44.50	49.22	15.03	46.08	38.76
459:416	477:433	55.82	52.49	7.76	30.00	42.27
478:434	497:452	27.35	53.02	1.88	10.00	26.88
498:453	517:472	39.87	72.24	4.33	32.50	38.63

Tabelle 5.8: Durchschnittliches F-Mass der automatischen Vorschläge für den Wirtschaftsteil von Smultron

Die Zahlen für den Wirtschaftsteil von Smultron fallen weniger ermutigend aus als diejenigen des Sofie-Teils. Die Ausbeute der Terminals liegt in ähnlichem Rahmen

zwischen 27% und 55%; ihre Präzision liegt zwischen 44% und 72%, also schon einiges niedriger. Ganz schlecht sieht die Ausbeute der Nonterminals aus: Sie liegt gerade mal zwischen 2% und 22%. Das F-Mass siedelt sich daher auch nur zwischen 22% und 45% an. Der Durchschnittswert der Präzision des Alignierungstyps liegt bei 76.78%.
Diese krassen Unterschiede lassen sich wohl, wie bereits erwähnt, mit der durchschnittlichen Satzlänge und der damit einhergehenden Komplexität der Syntaxstrukturen erklären. Die Sätze sind in der Sofie-Baumbank ziemlich kurz, während die Sätze des Wirtschaftsteils tendenziell lang sind. Da das DVD-Handbuch zwischen den beiden liegt, würde das hoffen lassen, dass die Werte dort wieder etwas besser werden. Als diese Arbeit geschrieben wurde, waren aber noch zu wenige Alignierungen vorhanden, um dies zu überprüfen. Zumindest die Präzisionswerte der Sofie-Baumbank sind zufriedenstellend; das erste Ziel war ja, eine hohe Präzision mit den automatischen Vorschlägen zu erreichen. Der richtige Alignierungstyp wird in beiden Teilen der Baumbanken sehr häufig getroffen. Wie die Ausbeute noch erhöht werden könnte, wird in Kapitel 6 noch andiskutiert werden.

Nun folgen ein paar praktische Beispiele, anhand derer die Alignierungsqualität diskutiert wird. Als Erstes schauen wir uns in Abbildung 5.8 die automatischen Alignierungen des Beispiels (3) (S.45) an, das sind die Sätze 522 (deutsch) und 521 (englisch). Als Vergleich dazu in Abbildung 5.9 noch die originalen Alignierungen.

Die Werte der automatischen Alignierung für das Baumpaar 522/521 sind folgende:

Ausbeute Terminals:	50%
Ausbeute Nonterminals:	50%
Präzision Terminals:	66%
Präzision Nonterminals:	100%
Präzision Alignierungstyp:	100%
F-Mass:	60%

Betrachten wir nun aber Abbildung 5.8: Die Pronomen und ihre direkten Elternknoten wurden richtig aligniert, ebenso wurde für *Wohnzimmer* eine Mehrwortalignierung zu *living room* erkannt. Fälschlicherweise wird *stand* mit *was* aligniert. Wie aber in der Original-Alignierung in Abbildung 5.9 zu sehen ist, gibt es für das deutsche Verb keine Entsprechung im englischen Satz. Daher lässt sich diese Fehlalignierung kaum verhindern, da die Werte für diesen Alignierungsvorschlag aufgrund von spärlichen Daten relativ hoch sind, wie wir bereits in Kapitel 5.2.1 diskutiert haben.

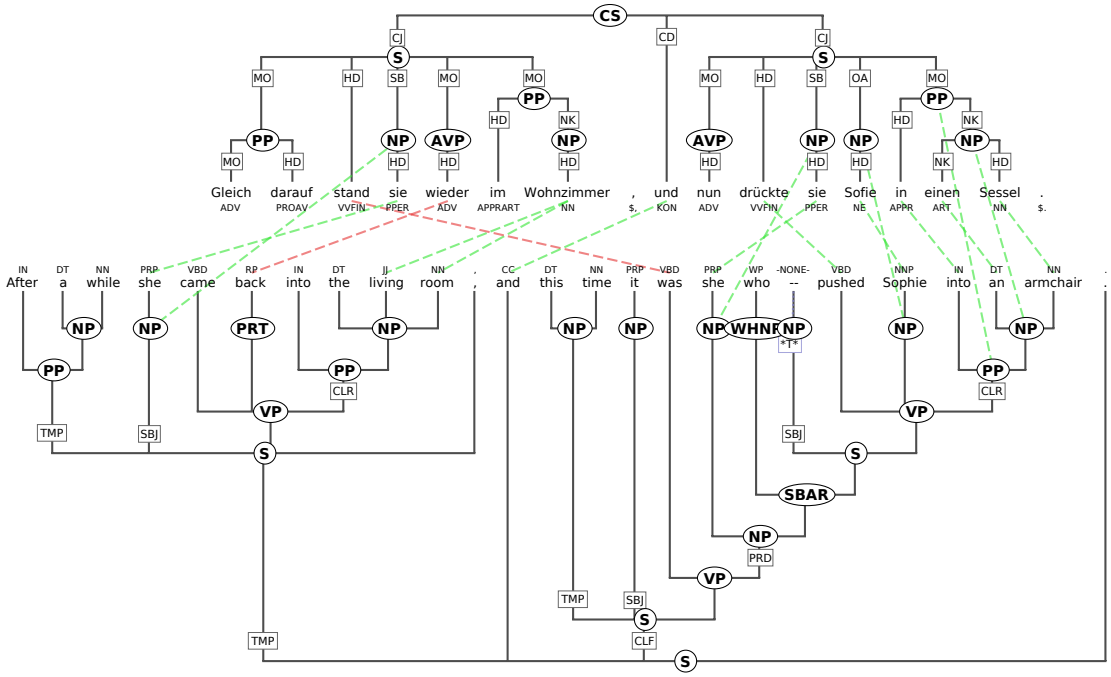


Abbildung 5.8: Beispiel (3), automatisch aligniert

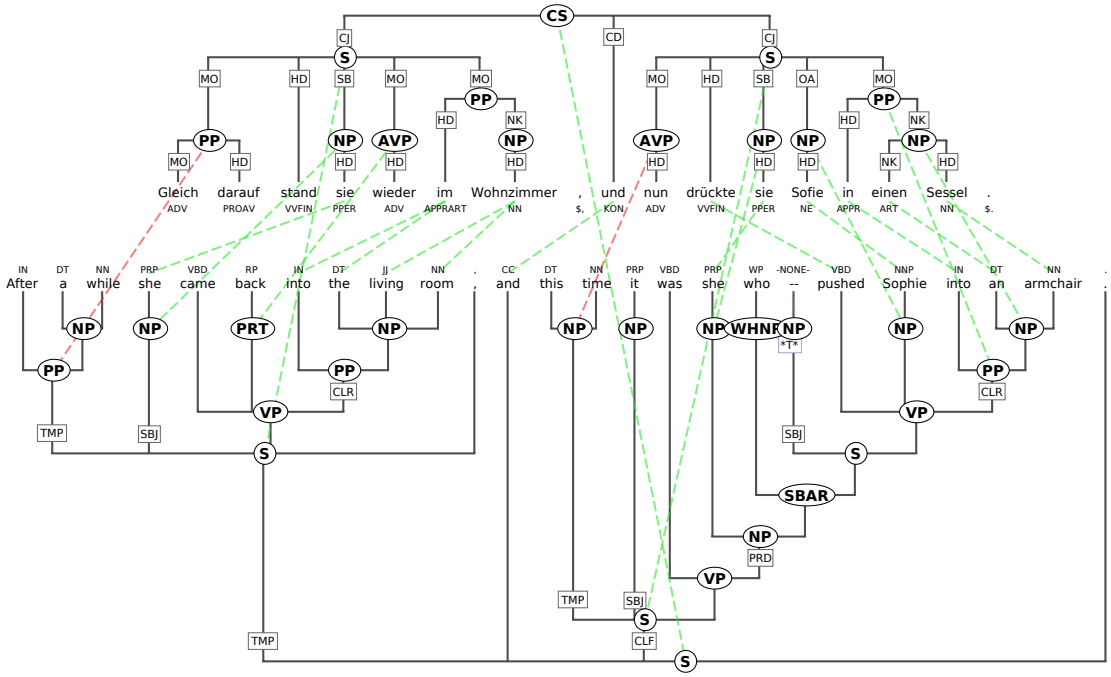


Abbildung 5.9: Beispiel (3), Original, manuell aligniert

Interessant ist die Entscheidung des Annotators, *wieder* nicht mit *back* zu alignieren, was automatisch vorgeschlagen wird, dafür aber ihre Elternknoten. Ich würde jedenfalls die automatische Alignierung nicht als Fehler ansehen, auch wenn diese Alignierung nur an einem einzigen anderen Ort in der Baumbank anzutreffen ist (Baumpaar de:s371 - en:s363). Das Fazit für dieses Satzpaar: Der Annotator müsste lediglich ein bis zwei falsche Alignierungen wieder löschen und zwei Terminal- und vier bis fünf Nonterminal-Alignierungen noch manuell einfügen.

Das nächste Beispiel ist gleich das folgende Baumpaar 523 (deutsch) und 522 (englisch). Die automatisch generierten Alignierungen sind in Abbildung 5.10 zu sehen.

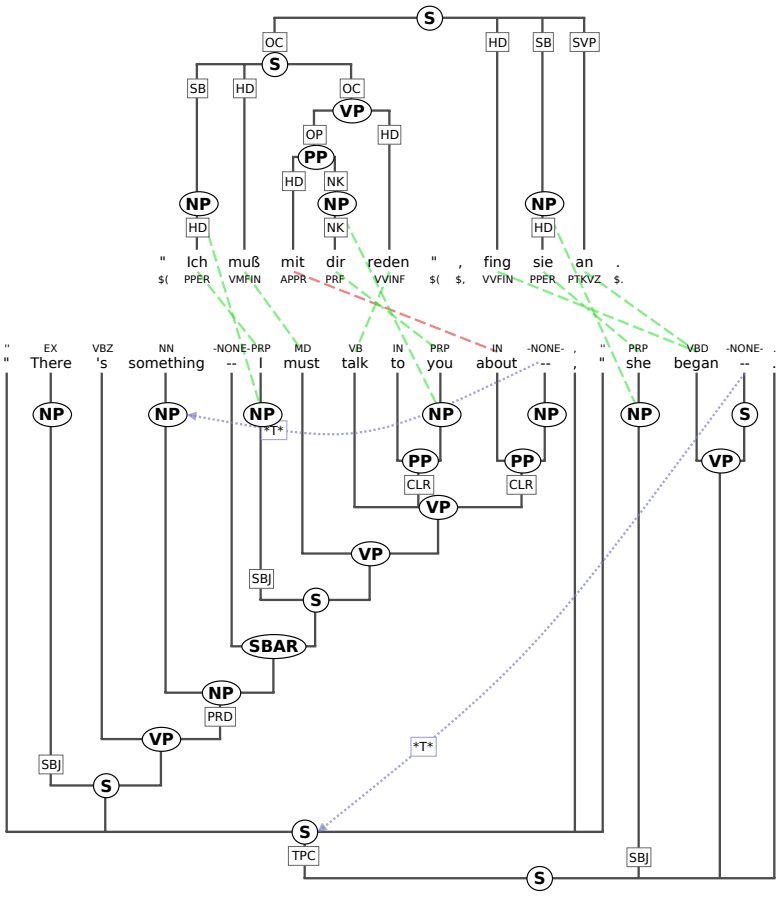


Abbildung 5.10: de:s523 - en:s522, automatisch aligniert

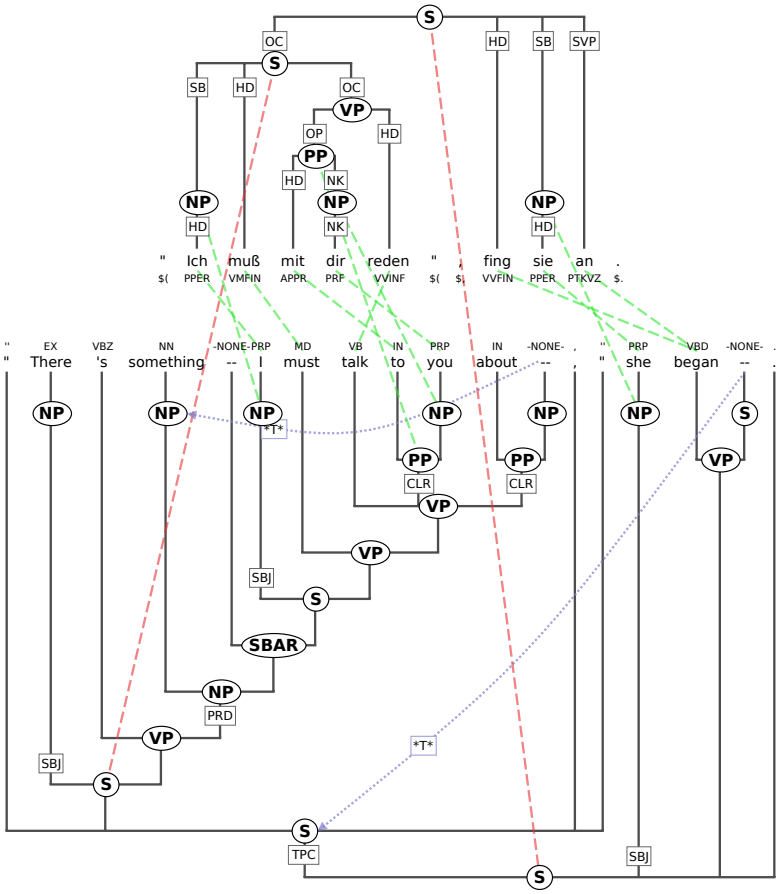


Abbildung 5.11: de:s523 - en:s522, Original, manuell aligniert

Die Werte der automatischen Alignierung für dieses Baumpaar:

Ausbeute Terminals:	87.5 %
Ausbeute Nonterminals:	50.0 %
Präzision Terminals:	87.5 %
Präzision Nonterminals:	100.0 %
Präzision Alignierungstyp:	100.0 %
F-Mass:	79.33%

Auch hier wurde eine Mehrwortalignierung richtig erkannt, nämlich diejenige von *fing an* zu *began*. Die Werte für Präzision und Ausbeute sind ja relativ hoch, einzig die Alignierung von *mit* zu *about* fällt negativ ins Auge. Ähnlich wie bei dem Fehler in Abbildung 5.8 liegt hier ein Problem mit spärlichen Daten (‘*sparse data*’) vor: Im Korpus ist *mit* grösstenteils mit *with* aligniert; *mit* und *about* ist zwei Mal aligniert, und die für unser Beispiel korrekte Alignierung zwischen *mit* und *to* kommt nur ein Mal vor, nämlich im Satzpaar de:s515-en:s514. Eine mögliche, jedoch aufwändige

Lösung könnte für solche Fälle sein, den näheren Kontext der Wörter zu betrachten: In beiden Fällen kommen die beiden Präpositionen *mit* und *to* in Zusammenhang mit *sprechen* und *talk* vor (*mit* - *about* hingegen ist nur im Zusammenhang mit *was ist* - *what* zu finden, das im aktuellen Satzpaar ja nicht vorhanden ist). Allerdings bleibt für einen solchen Ansatz das Problem, dass die Lemma-Information, welche für diesen Test notwendig wäre, in der englischen Baumbank nicht vorhanden ist.

Als letztes Beispiel möchte ich die Sätze 20 (deutsch) und 21 (englisch) aus dem Sofie-Teil von Smultron aufführen. Die automatischen Alignierungen sind in Abbildung 5.12 zu sehen. Das Augenmerk liegt hier auf den Alignierungen von *Sofie* mit *Sophie* und denjenigen zwischen den Pronomen *sie* und *she*. Die gefundenen Alignierungen stimmen mit denjenigen im Original zu 100% überein. So weit, so gut, doch Alignierungen dieser Art könnten insofern Probleme beim automatischen Alignieren verursachen, als dass mit *Sofie* - *Sophie* z.B. in zwei Teilsätzen eine sichere Alignierung gefunden wird, welche ansonsten jedoch nichts miteinander zu tun haben. Dies setzt also die Disambiguierungsstrategie 4 (siehe Kapitel 5.2.1, S.49) ausser Kraft. Auch vom linguistischen Standpunkt ist es fraglich, ob diese Nominalphrasen aligniert werden sollten, da sie zwar vom semantischen, jedoch nicht vom strukturellen Gesichtspunkt her übereinstimmen.

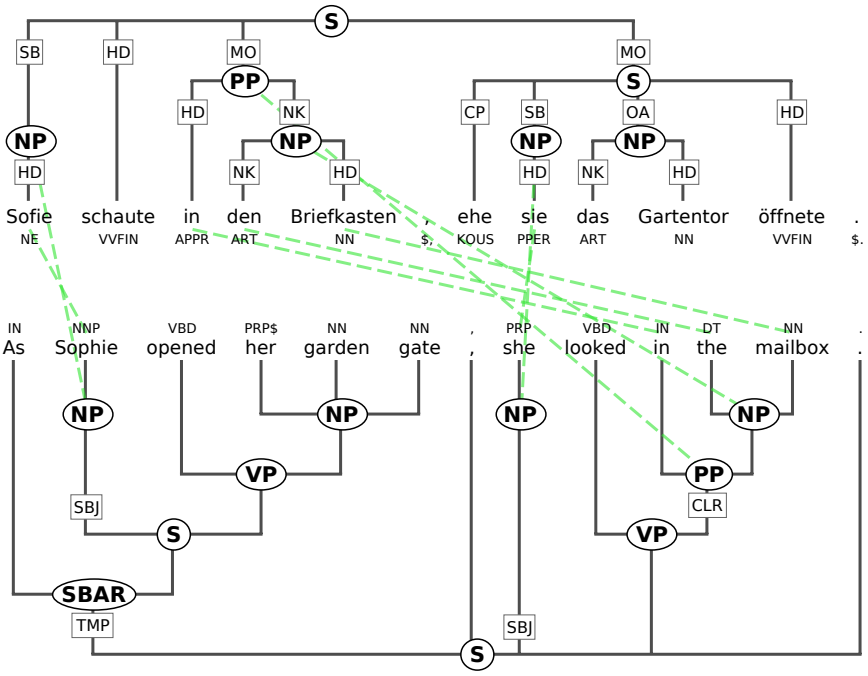


Abbildung 5.12: de:s20 - en:s21, automatisch aligniert

6 Rückblick, Fazit und Ausblick

Wir haben gesehen, was Baumbanken sind und wie sie erstellt werden, welche Suchwerkzeuge es zum Durchsuchen von Baumbanken gibt. Dann haben wir verschiedene Techniken von automatischen Wort-, Phrasen- und Satzalignierungen betrachtet und uns mit den Funktionalitäten des TreeAligners befasst. Schliesslich habe ich die Techniken zur automatischen Alignierung im TreeAligner vorgestellt und die Resultate evaluiert. Die Fragestellung, welche ich mir zu Beginn der Arbeit gestellt habe, wurde durch die Ergebnisse in Kapitel 5.4 teilweise beantwortet: Bereits mit ca. 250 alignierten Satzpaaren lassen sich Vorschläge generieren. Aus der Evaluation wurde ersichtlich, dass zumindest für kürzere Sätze die Präzision der automatischen Alignierungsvorschläge sehr gut ist. Also wurde in diesem Teilbereich das Ziel der Arbeit erreicht. Bei längeren Sätzen lässt die Präzision aber noch zu wünschen übrig, und was die Ausbeute betrifft, wäre ein höherer Wert sowohl im Sofie-Teil als auch im Wirtschaftsteil von Smultron anzustreben.

Zu implementieren sind unter anderem noch die Methoden, um weitere Alignierungen für Knoten zu finden, für welche kein Vorschlag gemacht werden konnte (siehe Kapitel 5.2.3, S.53, und Kapitel 5.3.3, S.60). Dafür könnten Ähnlichkeitsmasse, v.a. bei Eigennamen, Kompositazerlegung im Deutschen oder die Lemma-Suche miteinbezogen werden. Auch denkbar wäre es, für die Terminalalignierung ein externes Wörterbuch einzubinden; allerdings muss dafür natürlich für jedes Sprachpaar eine geeignete Ressource gefunden werden. Je nachdem könnte ein Wörterbuch auch wieder Ursache von ambigen Vorschlägen sein.
Auch bei der Nonterminalalignierung könnten allenfalls die Strategien erweitert werden. Gewisse Strategien ähneln einigen im Kapitel 3.3 (siehe S.25) vorgestellten Methoden: Z.B. vergleichen Samuelsson und Volk (2007) ebenfalls die Knotenlabels der Phrasen und die Kind-Terminalknoten mit N-Grammen. In meiner Methode gibt es keine vorgeschlagenen N-Gramme, aber ähnliche Daten aus den bereits vorhandenen Alignierungen. Mit den Methoden von Groves et al. (2004) gibt es gewisse Überschneidungspunkte, z.B. die Eltern- bzw. Kindknotenalignierung. Was man zum Testen auch für den TreeAligner anwenden könnte, wäre die Regel bei der Elternknotenalignierung: Wenn eine Phrase zwei Kindknoten hat, von denen

einer aligniert ist, werden sowohl die Phrasen als auch der Geschwisterknoten miteinander aligniert. Die anderen Methoden von Groves et al. (2004) sind jedoch schwierig anwendbar, da sie auf den Labels der Baumbanken beruhen, bzw. darauf, dass die Labels verglichen werden können. Da dies aber je nach Baumbank ändern kann und man daher manuell immer wieder neue Regeln schreiben müsste, scheint mir das schwer machbar zu sein. Die Methode von Tinsley et al. (2007) ist insofern vergleichbar mit derjenigen im TreeAligner, als dass auch sie in mehreren Durchläufen Hypothesen aufstellen und mit der Zeit anpassen.

Was von Anfang an als Möglichkeit geplant war, jedoch bis jetzt noch nicht umgesetzt wurde, ist das Einbinden der Alignierungen von anderen Projekten in die Alignierungs-Datenbank als Grundlage für ein neues oder noch spärlich aligniertes Projekt. Das könnte dem Problem der spärlichen Daten etwas entgegenwirken. Je nachdem, wie unterschiedlich die Textsorten der Projekte sind, könnte dies aber auch wenig bewirken, oder allenfalls sogar falsche Alignierungsvorschläge hervorbringen. Ein Versuch wäre auf jeden Fall sinnvoll.
Wenn erst wenige Alignierungen in einem (neuen) Projekt vorhanden sind, werden momentan kaum Vorschläge generiert, da die spärlichen Daten in Kombination mit einigen Disambiguierungsstrategien dies verhindern. Ein Lösungsansatz für dieses Problem wäre eventuell, Strategien wie den *mutual information score* und die Suche nach Alignierungen im selben Satzteil erst ab einer gewissen Grösse der Alignierungsdatenbank einzusetzen.
Es gibt also noch viele Möglichkeiten, die automatisch generierten Vorschläge im TreeAligner zu verbessern und so auch zum einfacheren Erstellen einer parallelen Baumbank einen guten Beitrag zu leisten. Zumindest die Grundlage dafür wurde in dieser Arbeit geschaffen, und die automatischen Vorschläge sind bereits in Gebrauch, um die Alignierungen des Handbuchteils von Smultron zu erstellen.

Glossar

ASCII ist die Abkürzung für American Standard Code for Information Interchange. ASCII ist ein numerischer Code für die verschiedenen Schriftzeichen (Buchstaben, Zahlen und einige Sonderzeichen), welcher gebraucht wird, um den Transfer von Information zwischen verschiedenen Maschinen und Programmen zu ermöglichen. Ausgehend vom ASCII-Code wurde ein umfangreicherer Code erschaffen, der Unicode. (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Dependenzgrammatik: Ein von Lucien Tesnière entwickeltes, am Strukturalismus orientiertes Modell zur Beschreibung der Syntax natürlicher Sprache. Das Hauptanliegen der Dependenzgrammatik ist die Beschreibung der Dependenz-Struktur eines Satzes, d.h. des Gefüges von Abhängigkeitsrelationen zwischen den Elementen eines Satzes. Dabei geht man davon aus, dass bei einer syntaktischen Verbindung zweier Elemente eines das regierende und das andere das abhängige Element ist. Wenn ein regierendes Element von einem anderen regierenden Element abhängig ist, dann entsteht eine komplexe hierarchische Dependenzordnung. Als Darstellung solcher Strukturen werden Baumgraphen verwendet, deren Zentralknoten das absolute "Regens" eines sprachlichen Gefüges ist (bei Sätzen das Verb). Die Abhängigkeitsrelation zu einem unmittelbar abhängigen Element wird durch eine Kante zu einem darunter stehenden Knoten dargestellt. (Quelle: H.Bussmann, *Lexikon der Sprachwissenschaft*)

Hidden Markov Model: Das Hidden Markov Model (HMM) ist benannt nach dem russischen Mathematiker A. A. Markov und ist wie ein endlicher Automat mit Ausgabe aufgebaut, bei dem aber Übergänge und Output auf Wahrscheinlichkeitsberechnungen beruhen. HMMs bestehen aus Zuständen, möglichen Übergängen zwischen diesen Zuständen und der Wahrscheinlichkeit des Eintreffens dieser Übergänge. In einem spezifischen Zustand kann ein Resultat generiert werden, indem alle Wahrscheinlichkeiten in Betracht gezogen werden. Nur das Resultat, nicht aber die Zustände, sind für einen externen Betrachter sichtbar. Die Zustände sind nach aussen verborgen (hidden). (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Kollokation: Als Kollokation bezeichnet man in der Linguistik charakteristische,

häufig auftretende Wortverbindungen, deren gemeinsames Vorkommen auf einer Regelhaftigkeit gegenseitiger Erwartbarkeit beruht, also primär semantisch (nicht grammatisch) begründet ist: z. B. Buch – dick, Tag – hell, Jesus – Christentum.

Lexikalisch-funktionale Grammatik (LFG): Die LFG wurde in den 70er Jahren von Joan Bresnan und Ronald Kaplan entwickelt. Im Gegensatz zur Syntax Chomskys, die durch Transformationen verbundene separate Ebenen der linguistischen Darstellung beinhaltet, basiert die LFG auf zwei sich gegenseitig einschränkenden Strukturen: einem Konstituentenbaum (C-Struktur, englisch *constituent*) und einer Merkmalsstruktur (F-Struktur, englisch *feature*). Die Konstituentenstruktur wird durch wenige, sehr allgemeine Phrasenstrukturregeln erzeugt, die mit Merkmalsgleichungen annotiert sind. Die funktionale Struktur ist eine Merkmalsstruktur, in der u.a. die grammatischen Funktionen (Subjekt, Objekt, Adjunkte usw.) als Merkmale repräsentiert sind. Sie wird durch Anwendung der Merkmalsgleichungen parallel zur Konstituentenstruktur aus den Merkmalsstrukturen der unmittelbaren Konstituenten aufgebaut. (Quellen: Wikipedia (`http://de.wikipedia.org`), CL-Glossar`http://www.cl.uzh.ch/kitt/clglossar`)

MySQL ist ein Relationales Datenbankverwaltungssystem, basierend auf der Datenbanksprache SQL. Es ist als Open-Source-Software für verschiedene Betriebssysteme verfügbar. (Quelle: Wikipedia, `http://de.wikipedia.org`)

Parsing bedeutet, Sprache zu analysieren, um ihre Struktur und die Beziehungen auf der Ebene der Morphologie, Syntax oder der Semantik darzustellen. Für die Analyse der Wortarten hat sich der Begriff Tagging etabliert, während Parsen über das blosse Annotieren eines Textes mit Wortarten hinausgeht. Ein Anwendungsgebiet der Computerlinguistik besteht darin, automatische Parser zu konstruieren. (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Nonterminal: Ein nicht-terminales Symbol ist ein Symbol, welches zur Strukturbeschreibung benutzt wird. In einer linguistischen Grammatik repräsentiert es eine lexikalische Kategorie oder eine Phrase. In einer lexikalischen Kategorie werden alle Worte, die zu einem spezifischen lexikalischen Symbol gehören, zusammengefasst. Alle Nomen werden zum Beispiel zur Kategorie N zusammengefasst. Eine Phrase fasst eine oder mehrere lexikalische Kategorien zusammen.

Beispiele für lexikalische Kategorien sind: N, V, ADJ, P.

Beispiele für Phrasen sind: NP, VP, PP, AdjP, S.

(Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Phrasenstrukturgrammatik: Ein Grammatiktyp des Amerikanischen Strukturalismus, der den syntaktischen Aufbau von Sätzen in Form von Konstituentenstrukturen, d.h. von hierarchisch geordneten Konstituenten beschreibt. [...] Dieser ursprünglich als Erkennungsgrammatik konzipierte Grammatiktyp erfährt im Rahmen der Generativen Syntax sowohl eine stärkere Formalisierung als auch eine teilweise Uminterpretation, insofern die ursprünglichen statischen, analytisch beschreibenden Regeln (z.B. $S \rightarrow NP + VP$) nunmehr zu generativen Ersetzungsregeln umgedeutet werden. Eine strikt an der Oberflächenstruktur operierende Phrasenstrukturgrammatik kann einer Reihe von syntaktisch-semantischen Problemen nicht hinreichend gerecht werden, z.B. diskontinuierlichen Elementen (wie *ablesen* in *Anne liest den Vortrag ab*), Mehrdeutigkeiten, u.a. (Quelle: H.Bussmann, *Lexikon der Sprachwissenschaft*)

POS bedeutet "part of speech". Darunter wird die Wortkategorie (lexikalische Kategorie) eines Wortes verstanden. Ein Wort hat in einem Satz die Form eines Substantivs, Verbs, Adjektivs, Adverbs, Pronomens oder einer Präposition etc., kann aber je nach Kontext eine andere Funktion übernehmen. (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Tagging: Beim Tagging wird jedem Token eines Satzes dessen Wortart (POS) hinzugefügt. Man bedient sich dazu einer definierten Menge von Tags aus einem sogenannten Tagset. Normalerweise werden auch den Satzzeichen Tags zugewiesen.

Tagging ist ein wichtiger Verarbeitungsschritt vor der syntaktischen Analyse, dem Parsing. Das zum Tagging verwendete Werkzeug ist ein Tagger. (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Terminal: Ein Terminal-Symbol einer Grammatik ist ein Symbol, welches die Zeichenketten des Alphabets der Grammatik beschreibt. In der Linguistik bedeutet dies, dass ein Terminal-Symbol ein Wort der Sprache repräsentiert, welche durch die Grammatik beschrieben wird. (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Token: Ein Token besteht aus einer Reihe von Buchstaben, deren Anfang und Ende durch einen Leerschlag oder durch ein anderes Begrenzungszeichen gekennzeichnet sind. Einzelne Token sind üblicherweise Wortformen oder Satzzeichen. Sie werden durch einen Tokenizer gefunden.

Tokens können in Beziehung zu Types gesetzt werden, wobei die Tokens jedes konkrete Vorkommen eines Zeichens bezeichnen, ein Type jedoch gleichlautende Tokens zu einer Klasse zusammenfügt. (Quelle: `http://www.cl.uzh.ch/kitt/clglossar`)

Literaturverzeichnis

Abeillé, Anne (2003): Treebanks: Building and Using Parsed Corpora, Band 20 von *Text, speech and language technology*, Kapitel Introduction, S. XII–XVI. Dordrecht: Kluwer Academic Publishers.

Ahrenberg, Lars (2007): *LinES: An English-Swedish Parallel Treebank*. In Proceedings of Nordiska Datalingvistdagarna, S. 270–274, Tartu, Estonia.

Banerjee, Satanjeev und Alon Lavie (2005): *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. In Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association for Computational Linguistics, S. 65–72, Ann Arbor, Michigan.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius und George Smith (2002): *The TIGER Treebank*. In Proceedings of the Workshop on Treebanks and Linguistic Theories, Sozopol.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra und Robert L. Mercer (1993): *The mathematics of statistical machine translation*. Computational Linguistics, 19(2):264–311.

Doddington, George (2002): *Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics*. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, S. 128–132, San Diego.

Dorna, Michael und Susanne Jekat (2001): *Maschinelle und computergestützte Übersetzung*. In Kai-Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde und Hagen Langer (Hrsg.) Computerlinguistik und Sprachtechnologie - Eine Einführung, S. 563–571, Heidelberg, Berlin: Spektrum Akademischer Verlag, 2. Auflage.

Gale, William A. und Kenneth W. Church (1993): *A programm for aligning sentences in bilingual corpora*. In Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL), S. 177–184.

Groves, Declan, Mary Hearne und Andy Way (2004): *Robust Sub-Sentential Alignment of Phrase-Structure Trees*. In Proceedings of Coling, S. 1072–1078, Geneva, Switzerland: COLING.

Göhring, Anne (2009): Spanish Expansion of a Parallel Treebank. Diplomarbeit, Universität Zürich.

Hearne, Mary und Andy Way (2003): *Seeing the Wood for the Trees: Data-Oriented Translation*. In MT Summit IX, S. 165–172, New Orleans: LO.

Kay, Martin und Martin Röscheisen (1993): *Text-Translation Alignment*. Computational Linguistics, 19:121–142.

König, Esther, Wolfgang Lezius und Holger Voormann (2003): TIGERSearch 2.1: User's Manual. IMS, Universität Stuttgart, Stuttgart.

Lezius, Wolfgang (2001): *Baumbanken*. In Kai-Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde und Hagen Langer (Hrsg.) Computerlinguistik und Sprachtechnologie - Eine Einführung, S. 414–422, Heidelberg, Berlin: Spektrum Akademischer Verlag, 2. Auflage.

Marek, Torsten (2009): Integration of Light-weight Semantics into a Syntax Query Formalism: An Extension of the Tiger Query Language. Diplomarbeit, Universität des Saarlandes.

Marek, Torsten, Joakim Lundborg und Martin Volk (2008): *Extending the TIGER Query Language with Universal Quantification*. In Text Resources and Lexical Knowledge (Proceedings of KONVENS), Berlin.

Mettler, Maël (2007): *Parallel Treebank Search - The Implementation of Stockholm TreeAligner Search*. C-Level Thesis.

Papineni, Kishore, Salim Roukos, Todd Ward und Wei-Jing Zhu (2002): *BLEU: a Method for Automatic Evaluation of Machine Translation*. In Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics, S. 311–318, Philadelphia, PA.

Rios Gonzales, Annette, Anne Göhring und Martin Volk (2009): *A Quechua-Spanish Parallel Treebank*. In Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories, S. 53–64, Groningen.

Rohde, Douglas L. T. (2002): TGrep2 User Manual.

Samuelsson, Yvonne und Martin Volk (2004): *Automatic Node Insertion for Treebank Deepening*. In Third Workshop on Treebanks and Linguistic Theories (TLT) 2004, Tübingen.

Samuelsson, Yvonne und Martin Volk (2006): *Phrase Alignment in Parallel Treebanks*. In Proceedings of the 5th Workshop on Treebanks and Linguistic Theories, Prag.

Samuelsson, Yvonne und Martin Volk (2007): *Automatic Phrase Alignment: Using Statistical N-Gram Alignment for Syntactic Phrase Alignment*. In Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories (TLT 2007), Bergen, Norway.

Samuelsson, Yvonne, Martin Volk und Sofia Gustafson-Čapková (2007): Alignment Guidelines for SMULTRON. Department of Linguistics, Stockholm University, Stockholm.

Skut, W., T. Brants, B. Krenn und H. Uszkoreit (1998): *A linguistically interpreted corpus of German newspaper texts*.
URL `citeseer.ist.psu.edu/skut93linguistically.html`

Taylor, Anne, Mitchell Marcus und Beatrice Santorini (2003): *The Penn Treebank. An Overview*. In Anne Abeillé (Hrsg.) Treebanks: Building and Using Parsed Corpora, Kapitel 1, S. 5–22, Dordrecht: Kluwer.
URL `http://www.cis.upenn.edu/~treebank/`

Tiedemann, Jörg (2003): Recycling Translations. Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing. Dissertation, Acta Universitatis Upsaliensis, Uppsala.

Tinsley, John, Ventsislav Zhechev, Mary Hearne und Andy Way (2007): *Robust Language Pair-Independent Sub-Tree Alignment*. In Machine Translation Summit XI Proceedings, Kopenhagen.

Vogel, Stephan, Hermann Ney und Christoph Tillmann (1996): *HMM-based word alignment in statistical translation*. In Proceedings of the 16th International Conference on Computational Linguistics (COLING), S. 836–841, Kopenhagen.

Volk, Martin und Yvonne Samuelsson (2004): *Bootstrapping Parallel Treebanks*. In Proceedings of the 5th International Workshop on linguistically Interpreted Corpora (COLING2004), S. 63–69, Geneva, Switzerland.

Zhechev, Ventsislav und Andy Way (2008): *Automatic Generation of Parallel Treebanks*. In Proceedings of the 22nd International Conference on Computational Linguistics, S. 1105–1112, Manchester.

A Tagsets

A.1 Penn-Treebank-Tagset

Dies ist die Version des Penn-Treebank-Tagsets, das in Smultron verwendet wurde. Es gibt wenige Abweichungen zum ursprünglichen Tagset: NNP entspricht NP, PRP entspricht PP, PRP$ entspricht PP$.

POS	**Beschreibung**	**Beispiel**
CC	Coordinating conjunction	*and, but, or, nor*
CD	Cardinal number	*[the] three [words]*
DT	Determiner	*a [person], the [supermarket]*
EX	Existential there	*[Is] there [life after death]*
FW	Foreign word	*P.S.*
IN	Preposition or subordinating conjunction	*at [some point], [she remembered] that*
JJ	Adjective	*[the] human [brain]*
JJR	Adjective, comparative	*more, better, older*
JJS	Adjective, superlative	*most, worst, quickest*
LS	List item marker	
MD	Modal	*[there] would [be], [I] cannot [expect]*
NN	Noun, singular or mass	*garden, everything, time*
NNS	Noun, plural	*houses, woods, people*
NNP	Proper noun, singular	*Sophie, Europe*
NPS	Proper noun, plural	*[in the] Americas, [Power Tecnology] Products*
PDT	Predeterminer	*[with] both [hands], such [a walk]*
POS	Possessive ending	*[Sohpie]'s [father]*
PRP	Personal pronoun	*he, she, it, you*
PRP$	Possessive pronoun	*[on] her [way], its [whiskers]*
RB	Adverb	*surely, almost, seldom*
RBR	Adverb, comparative	*[to be] more [precise], [no] longer*
RBS	Adverb, superlative	*[the] most [dangerous pesticides]*
RP	Particle	*[to make] up [for], [she jumped] up*
SYM	Symbol	*&*

POS	Beschreibung	Beispiel
TO	to	*[they continued] to [invest]*
UH	Interjection	*please, oh, drat*
VB	Verb, base form	*[made it] seem, [there would] be*
VBD	Verb, past tense	*[we] took, [she] had [walked]*
VBG	Verb, gerund or present participle	*[they had been] discussing, making [it nice]*
VBN	Verb, past participle	*[she had] walked, written [by hand]*
VBP	Verb, non-3rd person singular present	*[who] are [you], [I] think*
VBZ	Verb, 3rd person singular present	*[it]'s [impossible], [nobody] knows*
WDT	Wh-determiner	*whatever [he did], [everything] that [exists]*
WP	Wh-pronoun	*who, what*
WP$	Possessive wh-pronoun	*whose*
WRB	Wh-adverb	*when, where, how*

A.2 STTS-Tagset

POS	Beschreibung	Beispiel
ADJA	attributives Adjektiv	*[das] grosse [Haus]*
ADJD	adverbiales oder	*[er fährt] schnell*]
	prädikatives Adjektiv	*[er ist] schnell*
ADV	Adverb	*schon, bald, doch*
APPR	Präposition; Zirkumposition links	*in [der Stadt], ohne [mich]*
APPRART	Präposition mit Artikel	*im [Haus], zur [Sache]*
APPO	Postposition	*[ihm] zufolge, [der Sache] wegen*
APZR	Zirkumposition rechts	*[von jetzt] an*
ART	bestimmter oder unbestimmter Artikel	*der, die, das, ein, eine*
CARD	Kardinalzahl	*zwei [Männer], [im Jahre] 1994*
FM	Fremdsprachliches Material	*[Er hat das mit "]A big fish[" übersetzt]*
ITJ	Interjektion	*mhm, ach, tja*
KOUI	unterordnende Konjunktion mit "zu" und Infinitiv	*um [zu leben], anstatt [zu fragen]*
KOUS	unterordnende Konjunktion mit Satz	*weil, dass, damit, wenn, ob*
KON	nebenordnende Konjunktion	*und, oder, aber*
KOKOM	Vergleichspartikel, ohne Satz	*als, wie*
NN	Appellativa	*Tisch, Herr, das Reisen*
NE	Eigennamen	*Hans, Hamburg, HSV*
PDS	substituierendes Demonstrativpronomen	*dieser, jener*

POS	**Beschreibung**	**Beispiel**
PDAT	attribuierendes Demonstrativpronomen	*jener [Mensch]*
PIS	substituierendes Indefinitpronomen	*keiner, viele, man, niemand*
PIAT	attribuierendes Indefinitpronomen ohne Determiner	*kein [Mensch], irgendein [Glas]*
PIDAT	attribuierendes Indefinitpronomen mit Determiner	*[ein] wenig [Wasser], [die] beiden [Brüder]*
PPER	irreflexives Personalpronomen	*ich, er, ihm, mich, dir*
PPOSS	substituierendes Possessivpronomen	*meins, deiner*
PPOSAT	attribuierendes Possessivpronomen	*mein [Buch], deine [Mutter]*
PRELS	substituierendes Relativpronomen	*[der Hund,] der*
PRELAT	attribuierendes Relativpronomen	*[der Mann,] dessen [Hund]*
PRF	reflexives Personalpronomen	*sich, einander, dich, mir*
PWS	substituierendes Interrogativpronomen	*wer, was*
PWAT	attribuierendes Interrogativpronomen	*welche [Farbe], wessen [Hut]*
PWAV	adverbiales Interrogativ- oder Relativpronomen	*warum, wo, wann, worüber, wobei*
PAV	Pronominaladverb	*dafür, dabei, deswegen, trotzdem*
PTKZU	"zu" vor Infinitiv	*zu [gehen]*
PTKNEG	Negationspartikel	*nicht*
PTKVZ	abgetrennter Verbzusatz	*[er kommt] an, [er fährt] Rad*
PTKANT	Antwortpartikel	*ja, nein, danke, bitte*
PTKA	Partikel bei Adjektiv oder Adverb	*am [schönsten], zu [schnell]*
TRUNC	Kompositions-Erstglied	*An- [und Abreise]*
VVFIN	finites Verb, voll	*[du] gehst, [wir] kommen [an]*
VVIMP	Imperativ, voll	*komm [!]*
VVINF	Infinitiv, voll	*gehen, ankommen*
VVIZU	Infinitiv mit "zu", voll	*anzukommen, loszulassen*
VVPP	Partizip Perfekt, voll	*gegangen, angekommen*
VAFIN	finites Verb, aux	*[du] bist, [wir] werden*
VAIMP	Imperativ, aux	*sei [ruhig!]*
VAINF	Infinitiv, aux	*werden, sein*
VAPP	Partizip Perfekt, aux	*gewesen*
VMFIN	finites Verb, modal	*dürfen*
VMINF	Infinitiv, modal	*wollen*
VMPP	Partizip Perfekt, modal	*[er hat] gekonnt*
XY	Nichtwort, Sonderzeichen enthaltend	D2XW3
$,	Komma	,
$.	Satzbeendende Interpunktion	.?!;:
$(	sonstige Satzzeichen; satzintern	–, [], ()

B Evaluationsskript

Auf den nachfolgenden Seiten ist der Code des Skripts, welches ich für die Evaluation geschrieben habe. Das Skript lässt sich über die Kommandozeile starten, indem man die gewünschte Projektdatei als Argument mitgibt. Voraussetzung ist, dass das Paket Ladon referenziert wird. Als zweites Kommandozeilenargument kann optional noch ein Intervall in Form einer Ganzzahl mitgegeben werden, wenn nicht jeder, sondern nur jeder x. Satz aligniert werden soll. Die Klasse `ProjectManipulator` hat sieben Methoden. In der Methode `delete_align_loop` wird rückwärts (d.h. beginnend mit dem letzten alignierten Baumpaar der Baumbank) durch alle zu alignierenden Baumpaare iteriert. Zum Löschen der Alignierungen im aktuellen Baumpaar wird die Methode `delete_alignments` aufgerufen. Für das Alignieren des leeren Baumpaares ist danach die Methode `auto_align` zuständig. Als Nächstes wird `compare_to_goldstandard` aufgerufen. In dieser Methode werden die Resultate der automatischen Alignierung mit dem Goldstandard verglichen. Die Werte von Ausbeute, Präzision und F-Mass werden in eine Datei geschrieben. Am Schluss lässt sich mit `calculate_average` der Durchschnitt der Masse berechnen. Die letzten beiden Methoden `save_division` und `seperate_t_nt_pairs` sind Hilfsmethoden; erstere um Divisionen durch 0 zu verhindern, letztere um die Alignierungen als separate Listen für Terminal- und Nonterminalalignierungen zu erhalten.

```
# -*- coding: utf-8 -*-

import sys
import time

from ladon.align import types as align_types
from ladon.align.project import AlignmentProject
from ladon.align.auto_align.align_db import AlignDb
from ladon.align.auto_align import helper
from ladon.align.auto_align.alignment_generator import AlignmentGenerator

from ladon.utils.progress import ConsoleProgressMonitor

class ProjectManipulator(object):
    """ Class for evaluating automatic alignments in Ladon """
    def __init__(self):
        """
            The project file is command line passed by the first command line
argument.
            The second command line argument is optional: It's the interval for
choosing
            the sentences whose alignments are going to be deleted
        """
        self.project_filename = sys.argv[1]
        if len(sys.argv)==3:
            self.interval = sys.argv[2]
        else:
            self.interval = 1

        self.f_measures = []
        self.s_numbers = []
        self.terminal_recall = []
        self.terminal_prec = []
        self.nt_recall = []
        self.nt_prec = []

        self.unicode_filename= unicode(self.project_filename, "UTF-8")
        self.project = AlignmentProject.read_project(self.unicode_filename,
                       ConsoleProgressMonitor())
        self.project_title =  self.project.path.split("/")[-1].split(".xml")[0]

        self.evaluation_path = "../"+self.project_title+".eval"
        self.evaluation_file= open(self.evaluation_path,"w")
        self.evaluation_file.write(self.project_title)
        self.evaluation_file.write("\n \n")
        self.evaluation_file.write("Sentence pair \t Recall Ts \t Recall NTs \t
            Precision Ts \t Precision NTs \t Precision type\t F-Measure \n")

        # open the alignment project
        self.experimental_project = AlignmentProject.read_project(
                                    self.unicode_filename,
                                    ConsoleProgressMonitor())
        self.experimental_project._align_db = AlignDb(self.experimental_project)
        self.exp_align_index = self.experimental_project.align_index


    def delete_align_loop(self):
        """ Invokes the delete methods and the auto alignment """
        aligned_graphs = list(self.exp_align_index.aligned_graphs)
        last_pair = aligned_graphs[-1]
        last_graph1 = last_pair[0]
        last_sentence = last_graph1[1]
```

```
        start_deleting = last_sentence/2
        start_pair=[x for x in aligned_graphs if x[0][1]==start_deleting][0]
        start_index = aligned_graphs.index(start_pair)
        # the sentence pairs whose alignments have to be deleted and aligned
        delete_graphs = [aligned_graphs[g] for g in range(start_index,
                         len(aligned_graphs),self.interval)]
        delete_graphs.reverse()

        # since the list is reversed, the deletion of the alignments starts with
        # the last sentence pair. The auto alignments are not saved in the db,
        # but the deletions are, which means that the corpus of aligned
        # sentences is getting smaller while proceedings the loop
        for sentence_pair in delete_graphs:
            graph_id1 = sentence_pair[0]
            graph_id2= sentence_pair[1]
            self.delete_alignments(graph_id1, graph_id2)
            self.auto_align(graph_id1,graph_id2)
            self.compare_to_goldstandard(graph_id1, graph_id2)


    def delete_alignments(self,graph_id1, graph_id2):
        """ deletes all alignments in this sentence pair """
        align_list = self.exp_align_index.get_aligned_nodes(graph_id1,graph_id2)
        for (n1, n2) in align_list:
            self.exp_align_index.remove_alignment(n1, n2)
            self.experimental_project.align_db.remove_alignment((n1,n2),None)
            self.experimental_project.align_db.commit()


    def auto_align(self,graph_id1,graph_id2):
        """
            invokes the methodes for auto aligning the current sentence pair
        """
        graphs = dict((graph_id.corpus_id,
                 self.experimental_project.get_graph(graph_id)) for graph_id in
                 (graph_id1,graph_id2))
        ag = AlignmentGenerator(self.experimental_project, graphs)
        alignments_list = ag.align_clear_candidates()
        for a in alignments_list:
                source = a[1]
                target = a[2]
                info = align_types.User(type= a[3], author="auto",
checked=False, timestamp=time.localtime()[:3])
                self.exp_align_index.create_alignment(source, target, info)

    def compare_to_goldstandard(self,graph_id1,graph_id2):
        """
            The alignments that where created automatically are compared to the
            ones in the gold standard
        """
        gold_alignments =
            self.project.align_index.get_aligned_nodes(graph_id1,graph_id2)
            gold_pairs = set(gold_alignments.keys())

        auto_alignments =
            self.exp_align_index.get_aligned_nodes(graph_id1,graph_id2)
            auto_pairs =  set(auto_alignments.keys())
        inters_gold_auto =  auto_pairs & gold_pairs
        total_recall = self.save_division(len(inters_gold_auto),
                       len(gold_pairs)) * 100
        total_precision = self.save_division(len(inters_gold_auto),
                          len(auto_pairs)) * 100
```

```
        f_measure = self.save_division(2,(self.save_division(1,total_recall) +
                    self.save_division(1,total_precision)))

        graph1 = self.project.get_graph(graph_id1)

        gold_terminals, gold_nts = self.seperate_t_nt_pairs(gold_pairs,
                                   graph1,graph_id1.corpus_id)
        auto_terminals, auto_nts = self.seperate_t_nt_pairs(auto_pairs,
                                   graph1,graph_id1.corpus_id)

        inters_terminals, inters_nts =self.seperate_t_nt_pairs(inters_gold_auto,
                                      graph1,graph_id1.corpus_id)

        terminal_precision = self.save_division(len(inters_terminals),
                             len(auto_terminals)) * 100
        nt_precision = self.save_division(len(inters_nts),len(auto_nts)) * 100
        terminal_recall = self.save_division(len(inters_terminals),
                          len(gold_terminals)) * 100

        nt_recall = self.save_division(len(inters_nts) ,len(gold_nts)) * 100
        type_intersections = []

        for pair in inters_gold_auto:
            if auto_alignments[pair].type ==gold_alignments[pair].type:
                type_intersections.append(pair)

        type_precision = self.save_division(len(type_intersections),
                         len(inters_gold_auto)) * 100

        sentence_pair = '%s-%d : %s-%d' %(graph_id1.corpus_id,
                        graph_id1.ordinal, graph_id2.corpus_id,
                        graph_id2.ordinal, )
        line = '%s \t %d \t %d \t %d \t %d \t %d \t %d \n' %(sentence_pair,
                terminal_recall, nt_recall, terminal_precision, nt_precision,
                type_precision, f_measure)
        self.f_measures.append(f_measure)
        self.s_numbers.append(sentence_pair)
        self.terminal_recall.append(terminal_recall)
        self.terminal_prec.append(terminal_precision)
        self.nt_recall.append(nt_recall)
        self.nt_prec.append(nt_precision)
        self.evaluation_file.write(line)

    def seperate_t_nt_pairs(self,pairs,graph,corpus_id):
        """
            helper method for seperating the non termnal and terminal values
        """
        terminals= []
        non_terminals = []
        for n_id1, n_id2 in pairs:
            if n_id1[0]==corpus_id:
                id = n_id1[1]
            else:
                id = n_id2[1]
            if helper.non_terminal(id,graph):
                non_terminals.append((n_id1,n_id2))
            elif helper.terminal(id,graph):
                terminals.append((n_id1,n_id2))
        return terminals,non_terminals
```

```
    def save_division(self,a,b):
        """ Helper method to prevent division by 0"""
        if b ==0:
            return 0
        else:
            return float(a)/float(b)

    def calculate_average(self, zone, no_zeros):
        """
            Calculates the average of the recall, precision and f-measure
            values.
        """
        stars = "*"*50
        next_line = "\n \n %s Durchschnitt %s \n \n"%(stars,stars)
        self.evaluation_file.write(next_line)
        average_list = []
        i = zone
        f_len = len(self.f_measures)
        n_len = len(self.s_numbers)
        # if no_zeros is true, the entries where the f-measure is 0 are deleted
        #  in all lists
        if no_zeros:
            zero_len = len([y for y in self.f_measures if y==0])
            zero_pos = []
            for z in range(0,zero_len):
                zero_pos.append(self.f_measures.index(0,z))
            zero_pos = set(zero_pos)
            for zero in zero_pos:
                self.f_measures.pop(zero)
                self.terminal_recall.pop(zero)
                self.terminal_prec.pop(zero)
                self.nt_recall.pop(zero)
                self.nt_prec.pop(zero)
        # every xth entry the average of the zone is calculated
        for x in range(0,f_len,zone):
            i = x + zone
            if i + zone > f_len:
                i = f_len
            a = float(sum(self.f_measures[x:i]))/len(self.f_measures[x:i])
            trec_average = float(sum(self.terminal_recall[x:i]))/
                           len(self.terminal_recall[x:i])
            ntrec_average = float(sum(self.nt_recall[x:i]))/
                            len(self.nt_recall[x:i])
            tprec_average = float(sum(self.terminal_prec[x:i]))/
                            len(self.terminal_prec[x:i])
            ntprec_average = float(sum(self.nt_prec[x:i]))/
                             len(self.nt_prec[x:i])
            average_list.append(a)
            self.evaluation_file.write(self.s_numbers[x]+"/"+self.s_numbers[i-1]
                        + "\t" +str(trec_average)+ "\t"+ str(ntrec_average) +
                        "\t"+str(tprec_average)+ "\t"+str(ntprec_average) +
                        "\t"+str(a)+"\n")
            if i == f_len:
                break
        return average_list


pm = ProjectManipulator()
pm.delete_align_loop()
pm.calculate_average(20,False)
```

Lebenslauf

Persönliche Angaben

Name:	Sandra Roth
Adresse:	Blumenfeldstrasse 5
	8046 Zürich
Tel.:	043 811 37 73
Mailadresse:	`sandra@wgwh.ch`
Geboren am:	4. November 1980
Zivilstand:	ledig

Schulbildung

1987–1993	Primarschule Oberglatt ZH
1993–1995	Sekundarschule Rümlang ZH
1995–2000	Kantonsschule Bülach ZH, Typus D
seit 2000	Studium der italienischen Sprach- und Literaturwissenschaft, Computerlinguistik und deutschen Linguistik an der Universität Zürich

Berufliche und nebenberufliche Tätigkeiten

2000–2003	Mitarbeiterin Administration/Einkauf bei Leuenberger AG, Oberglatt
seit 2004	Hilfsassistentin am Institut für Computerlinguistik
2004–2009	Vorstandsmitglied des Fachvereins für Computerlinguistik, ab 2006 Präsidentin
2009/2010	Mitglied im OK für die TaCoS (Tagung der Computerlinguistik Studierenden) 2010 in Zürich

Hobbies

Musik:	Saxophonistin in der Jason Boon Bigband
	Klarinettistin in der Polyband Zürich
Sport:	Volleyball