Diplomarbeit

Robuste korpusbasierte Maschinelle Übersetzung mit Translation Templates

Ralf Engel

Universität des Saarlandes
Lehrstuhl Prof. Dr. W. Wahlster

18. Januar 1999

Ralf Engel
Waldhausweg 7
66123 Saarbrücken
ralf.engel@dfki.de

Diese Diplomarbeit wurde geschrieben unter Betreuung von Dr. Norbert Reithinger. Ich möchte mich bei ihm und weiterhin auch bei Michael Kipp und Jan Alexandersson für hilfreiche Tips und geduldiges Korrekturlesen bedanken.

Hiermit erkläre ich an Eides statt, daß ich zur Anfertigung dieser Arbeit nur die in der Arbeit angegebenen Hilfsmittel verwendet habe.

gez. Ralf Engel

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ziel der Arbeit

Bei der Maschinellen Übersetzung von natürlicher Sprache sind komplexe Probleme zu lösen. Hauptprobleme sind, die syntaktische Struktur des zu übersetzenden Satzes richtig zu erkennen, unter verschiedenen Übersetzungsvarianten die richtige auszuwählen und eine verständliche, sprachlich korrekte Übersetzung zu generieren. Erschwerend kommt hinzu, daß für die Auswahl der korrekten Übersetzung häufig Weltwissen notwendig ist bzw. der Text inhaltlich erfaßt und interpretiert werden muß. Zusätzliche Schwierigkeiten bereitet die enorme Datenmenge im sprachlichen Bereich, die kodiert werden muß.

Verschiedene Lösungsansätze bieten sich an, angefangen von Systemen mit einfacher Analyse und Verarbeitung bis hin zu sehr komplexen Systemen mit integrierter semantischer Verarbeitung.

Das System, das in dieser Arbeit präsentiert wird, basiert auf einem bewußt einfach gehaltenen und robusten beispielbasierten Ansatz mit flacher syntaktischer Analyse. Robustheit ist notwendig, da gesprochene Sprache verarbeitet wird, die häufig grammatikalisch inkorrekt ist und Worterkennungsfehler beinhalten kann. Der unkomplizierte Aufbau ermöglicht kurze Verarbeitungszeiten, die erforderlich sind, da das System online übersetzen muß. Der beispielbasierte Ansatz ermöglicht einen raschen Aufbau der sprachlichen Wissensbasis. Bei dem großen Umfang der Wissensbasis ist dies vorteilhaft.

Ziel dieser Arbeit ist die Implementierung eines solchen prototypischen beispielbasierten deutsch-englischen Übersetzungssystems im Rahmen des Verbmobil-Verbundprojektes [Wah93], [BKW96]. Dabei konnte auf die Vorarbeiten in diesem Projekt zurückgegriffen werden, insbesondere auf einen vorhandenen bilingualen Korpus.

Die syntaktische Verarbeitung bei meinem System beschränkt sich auf die Erkennung von Nominal- und Präpositionalphrasen sowie Nebensatzgrenzen. Die Basis der syntaktischen Analyse bilden hierarchische endliche Automaten und ein entsprechendes Verarbeitungsprogramm. Dieses Programm ist so ausgelegt, daß es

vielseitig angepaßt und erweitert werden kann, da hierarchische Automaten in vielen Bereichen eingesetzt werden können.

Grundlage des Übersetzungsprozesses bilden Translation Templates, die aus dem Korpus automatisch extrahiert werden. Diese bestehen aus einem ziel- und einem ausgangssprachlichen Teilbaum. Als Kontext für die Übersetzung wird jedem Template das Satzpaar beigefügt, aus dem es extrahiert wurde.

Bei der Übersetzung wird der Syntaxbaum für den ausgangssprachlichen Satz aus einzelnen im Beispielkorpus vorkommenden ausgangssprachlichen Teilbäumen nachgebildet. Im Unterschied zu anderen Systemen dürfen sich die Teilbäume überlappen, solange die Wörter der Teilbäume disjunkt sind. Dies ermöglicht einen Übersetzungsprozeß, der relativ unabhängig von der Struktur des ausgangs- bzw. zielsprachlichen Syntaxbaums ist. Als Auswahlkriterium zwischen verschiedenen Translation Templates wird der beigefügte Kontext eingesetzt, um die geeignetste Übersetzung auszuwählen. Aus den zielsprachlichen Teilbäumen wird der komplette zielsprachliche Syntaxbaum zusammengesetzt. Dieser wird mittels hierarchischer Automaten linearisiert.

Bei der Übersetzungskomponente in meinem System ist sprachliches Wissen weitgehend vom Programmcode getrennt. Bei der Entwicklung dieser Komponente habe ich Wert darauf gelegt, daß das Programm an neue Aufgaben angepaßt werden kann, ohne die Grundstruktur verändern zu müssen. Damit kann das Programm allgemein zu kontextabhängigen Transformationen von Bäumen verwendet werden.

Die internen Vorgänge beim Übersetzen durch eine graphische Oberfläche möglichst transparent darzustellen, war ein weiteres Ziel.

Die weitere Arbeit ist wie folgt aufgebaut:
Nach einer kurzen Einführung in die Maschinelle Übersetzung werden vorhandene prototypische beispielbasierte Systeme vorgestellt (Kapitel 2). Kapitel 3 stellt mein System im Überblick dar. In den Kapiteln 4-7 werden die einzelnen Komponenten des Systems (syntaktische Analyse, Template-Generierung, Transfer und Generierung) detailliert beschrieben. Kapitel 8 enthält eine Evaluation des Systems, wobei ein Teil der vorhandenen Beispieldialoge mittels leave-one-out getestet wurde. Eine Zusammenfassung und ein Ausblick auf künftige mögliche Erweiterungen findet sich in Kapitel 9.

1.2 Maschinelle Übersetzung im Überblick

Zur besseren Einordnung meines Systems wird erst ein Überblick über die unterschiedlichen Ansätze bei der Maschinellen Übersetzung gegeben. Eine gute Einführung in die Maschinelle Übersetzung geben die Bücher [HS92],[NCTG92] und [ABH+94].

Bei fast allen Übersetzungssystemen kann eine Einteilung in drei nacheinander zu durchlaufende Module vorgenommen werden.

Analyse Der Ausgangssatz wird morphologisch und syntaktisch analysiert. Bei einigen Systemen (z.B. VERBMOBIL) kommt außerdem noch eine semantische Analyse hinzu.

Transfer Die von der Analyse aufgebaute ausgangssprachliche Struktur wird in eine zielsprachliche Struktur transferiert, die die Generierung verarbeiten kann.

Generierung Aus der von der Transferkomponente gelieferten zielsprachlichen Struktur wird ein natürlichsprachlicher Satz in der Zielsprache generiert.

Ein Hauptunterschied zwischen den einzelnen Übersetzungssystemen liegt im Abstraktionsgrad der Strukturen, die zwischen den Modulen ausgetauscht werden. Das Spektrum umfaßt Strukturen, die mit einfacher syntaktischer Analyse erzeugt werden, sowie Strukturen, die eine sprachnahe semantische Repräsentation darstellen (wie z.B. in VERBMOBIL) bis hin zu Strukturen, die auf sprachunabhängiger Basis arbeiten (auch Interlingua genannt). Im allgemeinen gilt: je abstrakter die übergebenen Strukturen sind, desto geringer ist der Aufwand für die Transferkomponente, natürlich bei gleicher Übersetzungsleistung. In Abb. 1.1 ist dieser Sachverhalt veranschaulicht.

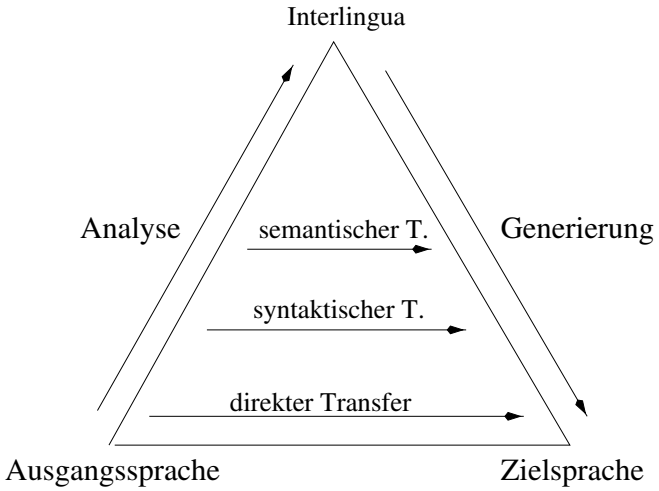

Abbildung 1.1: Das Verhältnis des Aufwandes bei Analyse, Transfer und Generierung (entnommen [HS92])

Der Vorteil von Übersetzungssystemen mit einer aufwendigeren Analyse- und Generierungskomponente und dafür einer weniger aufwendigen Transferkomponente ist, daß bei multilingualen Übersetzungssystemen mit n verschiedenen Sprachen für jede Sprache nur ein Analyse- und ein Generierungsmodul, aber $n(n-1)$ Transfermodule gebaut werden müssen (s.a. Abb. 1.2). Bei dem theoretischen Idealfall würde die Transferkomponente ganz entfallen und das Analysemodul eine Struktur erzeugen, die das Generierungsmodul direkt verarbeiten kann. Hier

ist dann die Transferleistung in die Analyse- bzw. Generierungsmodule integriert. Ein höheres abstraktes Niveau erlaubt zudem noch eine semantische Verarbeitung, die u.a. bei ambiger Bedeutung eines Satzes Weltwissen hinzuziehen und Anaphern auflösen kann.

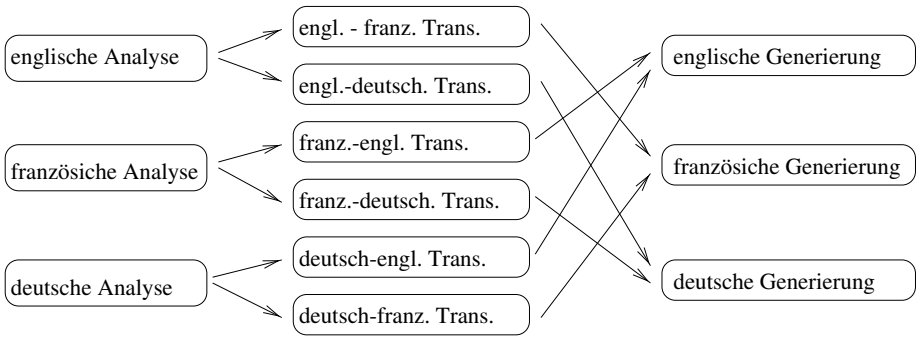

Abbildung 1.2: n*(n-1) Transferkomponenten werden benötigt (entnommen [HS92])

Ein weiteres Unterscheidungsmerkmal zwischen den einzelnen Übersetzungssystemen ist, inwieweit satzübergreifend übersetzt wird, d.h. ob und wie stark die Sätze, die vor und nach dem zu übersetzenden Satz stehen, bei der Übersetzung berücksichtigt werden. Vor allem bei der Übersetzung von Ellipsen ist dies häufig unverzichtbar. Eng damit gekoppelt ist, ob und wie detailliert festgestellt wird, in welche Domäne der zu übersetzende Satz fällt, z.B. ob es sich um einen Wirtschaftsbericht handelt.

Außerdem kann bei Übersetzungssystemen unterschieden werden, ob im Transfermodul Regeln (RBMT – rule based machine translation) oder Beispiele (EBMT – example based machine translation) zum Einsatz kommen. Regeln müssen dabei manuell von einem Spezialisten erstellt werden, während Beispielübersetzungen entweder ebenfalls manuell zusammengestellt werden oder aus alignierten bilingualen Korpora stammen können. Die Grenzen zwischen beiden Ansätzen sind dabei allerdings fließend. So gibt es Systeme, die zwar auf Regeln basieren, aber zur Auswahl der passendsten Regel Beispiele einsetzen oder Beispiele nur in Teilbereichen einsetzen, z.B. für Nominalphrasen.

1.3 Beispielbasierte Systeme

Da es sich bei meinem System um ein beispielbasiertes System handelt, wird auf diesen Ansatz noch näher eingegangen. Regelbasierte Systeme werden hingegen nicht detailliert vorgestellt.

Wesentliche Unterscheidungsmerkmale zwischen den einzelnen EBMT-System sind:

Tiefe der Verarbeitung Auch bei der beispielbasierten Übersetzung reicht das Spektrum von fehlender syntaktischer Analyse bis hin zu sehr aufwendiger syntaktischer Analyse.

Abstraktionsfähigkeit Um eine gute Abdeckung bei der Übersetzung zu erzielen, müssen die vorhandenen Beispielsatzpaare abstrahiert werden können, d.h. Abweichungen in dem zu übersetzenden Text gegenüber den Beispielsatzpaaren müssen möglich sein.

Korpora-Verarbeitung Die einzelnen Systeme unterscheiden sich darin, wie groß die Korpora sein dürfen, ob sie gut mit fehlerbehafteten Korpora umgehen können, wie frei die Übersetzungen sein dürfen und ob manuelle Änderungen am Korpus nötig sind.

Auf die letzten beiden Punkte soll detaillierter eingegangen werden:

Abstraktionsfähigkeit

Ein wichtiges Unterscheidungskriterium zwischen einzelnen EBMT-Systemen ist, in welchem Umfang die gegebenen Beispiele abstrahiert werden können. Bei einfachen Systemen, vor allem bei reinen *Translation Memories*, die nur eine Übersetzungshilfe darstellen, muß die Übereinstimmung zwischen dem zu übersetzenden Satz und dem Beispielsatz recht groß sein. Häufig muß sogar die Wortreihenfolge übereinstimmen. Anspruchsvollere Systeme kommen mit veränderter Wortstellung und morphologischen Unterschieden zurecht, z.B. können sie Singular in Plural umwandeln oder die Fälle entsprechend anpassen. Eine weitere Frage ist, ob mit zusätzlichen bzw. fehlenden Worten (z.B. Adjektive in Nominalphrasen) umgegangen werden kann.

Die Systeme unterscheiden sich darüber hinaus noch, in welchem Umfang Beispielsätze strukturelle Umformungen im Zielsprachensatz veranlassen können. So muß bei manchen Verben beim Übersetzen Subjekt und Objekt vertauscht werden oder aus einer Präpositionalphrase ein Objekt generiert werden, z.B.

sich treffen mit jdn. $\rightarrow$ *to meet sb.*

Eine noch aufwendigere Umformung stellen Wortartwechsel dar. Dafür ein Beispiel:

gern etw. machen $\rightarrow$ *to like to do sth.*

Das Problem dabei ist, ob so eine Umformung erkannt und abstrahiert werden kann, d.h. ob wenige Beispielsätze ausreichen, um zu erkennen, daß grundsätzlich eine Umformung notwendig ist. So muß bei *treffen* festgestellt werden, daß aus der *mit*-Präpositionalphrase immer ein indirektes Objekt wird.

Korpora-Verarbeitung

Bei beispielbasierten Systemen kann zudem noch unterschieden werden, ob sie direkt umfangreiche, weitgehend unbearbeitete bilinguale Korpora verwenden

können oder ob speziell ausgewählte, wenig redundante Beispielsätze manuell eingegeben werden müssen. Systeme, die mit großen Korpora umgehen können, sind natürlich im Vorteil, wenn bereits ein solcher Korpus vorhanden ist. Das ist bei technischen Dokumentationen häufig der Fall. In diesem Fall kann das System mit geringem Aufwand in neuen Domänen eingesetzt werden. Wie überzeugend dies in der Praxis funktioniert, hängt stark vom Korpus ab, besonders wie frei der Text übersetzt wurde. Wenn zu frei übersetzt wurde, ist die Gefahr sehr groß, daß nicht geeignete Beispielsätze ausgewählt werden. Durch die große Anzahl der Sätze ist es relativ aufwendig, den Korpus manuell zu überarbeiten. In diesem Fall kann es günstiger sein, die Sätze manuell unter zu Hilfenahme des Korpus zu erstellen. Eine andere Möglichkeit ist, die Sätze mit Hilfe eines zusätzlichen Tools zu filtern und dann noch manuell nachzubearbeiten.

Vor- und Nachteile

Die prinzipiellen Vorteile von beispielbasierten Systemen gegenüber regelbasierten Ansätzen sind nachfolgend dargestellt (siehe auch [Nom92]):

- Korrespondenz und technische Dokumentation in Firmen erfordert meist ein speziell angepaßtes Übersetzungssystem mit entsprechendem Wortschatz und Formulierungen. Dies geht einfacher und schneller – und somit auch kostengünstiger – mit beispielbasierten Systemen. Hinzu kommt, daß weniger systeminternes Wissen notwendig ist, z.B. wie Regeln erstellt werden und was dabei zu beachten ist. Dadurch wird die Einarbeitungszeit der Mitarbeiter erheblich verkürzt.

- Änderungen im laufenden Betrieb können einfacher und mit weniger Spezialwissen durchgeführt werden.

- In vielen Domänen, z.B. bei technischen Dokumentationen, kann auf bestehende Übersetzungen zurückgegriffen werden.

- Die Verzahnung von sprachlichen Daten und dem eigentlichen Übersetzungsalgorithmus ist nicht so eng, somit ziehen Änderungen am Programm keine so umfangreichen Änderungen an den sprachlichen Daten nach sich wie bei regelbasierten Systemen.

- Die Erstellung freier Übersetzungen für längere Sätze ist leichter möglich, wobei diese Übersetzungen – je nach Übersetzungssystem – auch noch bei kleineren Abweichungen funktionieren. Dies ermöglicht eine höhere Übersetzungsqualität, vor allem bei Texten, die eine hohe Redundanz aufweisen.

Allerdings muß sich in der Praxis zeigen, in welchem Ausmaß die aufgeführten prinzipiellen Vorteile sich tatsächlich auswirken:

- Um die Anzahl der Beispielsätze in einem vertretbaren Rahmen zu halten, müssen Beispielsätze verallgemeinert werden. Inwiefern und wie stark ein Satz verallgemeinert werden kann, ist allerdings extrem schwierig herauszufinden. Falsche Verallgemeinerungen führen jedoch in den meisten Fällen

zu nicht korrekten Übersetzungen. Bei regelbasierten Übersetzungen hingegen ist normalerweise in den Regeln kodiert, wie weit sie verallgemeinert werden können (z.B. mit Hilfe von Constraints).

- Die Auswahlfunktion, die zwischen verschiedenen Beispielsatzpaaren mit gleichem ausgangssprachlichen Beispielsatz entscheidet, ist ebenfalls sehr schwierig zu entwerfen und zu ändern. Dies liegt daran, daß sie global arbeitet, d.h. alle Beipsielsätze sind von Änderungen betroffen. Entsprechend schwierig gestalten sich dann Änderungen bzw. Erweiterungen.

- Ein höherer Resourcenverbrauch ist erforderlich, da Regelsammlungen i.a. kompakter sind als Beispielsammlungen. Dies ist bei Übersetzungssystemen, die ohnehin einen großen Resourcenbedarf erfordern, nachteilig.

1.4 Stand der Forschung

Bevor mein System detailliert beschrieben wird, werden in Kapitel 2 einige beispielbasierte Systeme ausführlich vorgestellt. Vorab ein kurzer Überblick über die vorgestellten Systeme:

- Ein System, das am **ATR** in Japan entwickelt wird. Es handelt sich dabei um ein regelbasiertes System, bei dem die Auswahl von alternativen Übersetzungsregeln anhand von Beispielen getroffen wird. Allerdings sind die Regeln so einfach aufgebaut, daß sie prinzipiell auch automatisch erstellt werden können, was allerdings bislang nicht realisiert wurde.

- Ein Vorschlag, wie solche Regeln aus bilingualen Korpora extrahiert werden können, wurde von einem Hitachi-Forschungslabor entwickelt (Hitachi-Template-Generierung, **HTG**).

- Einen völlig anderen Ansatz verfolgt die beispielbasierte Komponente des **Pangloss-Systems**, das große Korpora (270MB) verwendet, dabei ganz ohne syntaktische Analyse auskommt und deshalb nur geringe Abstraktionsfähigkeit besitzt. Das Pangloss-System wurde gemeinsam von mehreren amerikanischen Universitäten entwickelt und enthält auch eine regelbasierte Komponente.

- Zwei Systeme werden ebenfalls kurz vorgestellt, die zwar keine beispielbasierten Systeme sind, aber von denen einige Ideen übernommen wurden. Der zugrundeliegende Algorithmus eignet sich somit prinzipiell für beispielbasierte Übersetzung. Es handelt sich um das **Shake-and-Bake**-Übersetzungsverfahren und die **STAG**-Übersetzung, die auf dem Grammatikformalismus TAG (Tree Adjoining Grammar) aufbaut.

Außer den beschriebenen Systemen wurde noch ein dem ATR-System ähnlicher Prototyp von Satoshi Sato und Makoto Nagao an der Kyoto University entwickelt [SN90].

Einige Einzelaspekte der beispielbasierten Maschinellen Übersetzung werden in der Fachliteratur beschrieben, auf die hier aber nicht detaillierter eingegangen wird. Dazu eine kurze Aufzählung: So befaßt sich ein Artikel mit der Unterscheidung bei der Alignierung zwischen allgemeinen Fällen und Spezialfällen, die nicht so stark verallgemeinert werden dürfen [Wat94]. Eine weitere Arbeit behandelt das Matchen von Syntaxbäumen zwischen dem zu übersetzenden Satz und dem ausgangssprachlichen Teil des Beispielsatzpaares [MW92]. In [Nom92] wird darauf eingegangen, wie die Beispielsatzpaare sinnvoll zusammengefaßt werden können und somit deren Anzahl verringert werden kann. [HM97] beschäftigt sich mit der Auswahl des richtigen Beispielsatzpaares anhand von *decision trees*.

Kapitel 2

Existierende Systeme

2.1 TBMT am ATR

Am ATR in Koyoto, Japan wurde ab 1991 von Osamu Furuse, Hitoshi Iida und Eiichiro Sumita ein transferbasiertes maschinelles Übersetzungssystem (TBMT – transfer based machine translation) entwickelt [SI91], [FI92a], [FI92b], [FI94], [SI95], [ISF96], bei dem die Auswahl zwischen alternativen Transferregeln anhand von Beispielen und nicht durch Regeln erfolgt, deren Erstellung detaillierte Kenntnisse des Systems voraussetzen. Dieses System existiert als Prototyp, übersetzt Japanisch-Englisch und Englisch-Japanisch und kann auch gesprochene Eingabe verarbeiten.

Die einzelnen Verarbeitungsschritte sehen dabei folgendermaßen aus:

- Parsen des Ausgangssatzes mit einem auf Templates basierendem Chartparser. Die Templates haben dabei meist die Form *X Funktionswort Y*, also z.B. *X of Y* und müssen manuell erstellt werden. Durch rekursive Anwendung der Templates entsteht dabei ein Syntaxbaum.
- Für jedes ausgangssprachliche Template existiert mindestens ein entsprechendes Template in der Zielsprache zusammen mit Beispielen, die die Auswahl bei mehreren zielsprachlichen Templates ermöglichen. Der ausgangssprachliche Syntaxbaum wird nun in einen zielsprachlichen Syntaxbaum transferiert, indem jedes ausgangssprachliche Template durch ein zielsprachliches Template ersetzt wird.
- Die Generierung des Satzes in der Zielsprache erfolgt, indem der Parsebaum durchlaufen wird und die zielsprachlichen Templates zu einem natürlichsprachlichen Satz zusammengefügt werden.

Anhand des japanischen Satzes

> *kaigi*{*conference*} *no*{*of*} *toorokuryou*{*registration fee*} *wa*{*subject-marker*} *annaisho*{*announcement*} *ni*{*in*} *kisaisa re teimasu*{*be listed*}

dessen englische Übersetzung

The conference registration fee is listed in the announcement

ist, werden die einzelnen Schritte im Detail erörtert (entnommen [SI95]).

Syntaktische Analyse

Der Ausgangssatz wird mit Hilfe manuell erstellter Templates zerlegt. Die Templates, die zur Analyse des oben erwähnten Beispielsatzes benötigt werden, sind:

X no Y, X wa Y, X ni Y, X re und *X teimasu*

Damit läßt sich dann der in Abb. 2.1 dargestellte Syntaxbaum aufbauen. In diesem Fall ist das der einzig mögliche Syntaxbaum, da die Reihenfolge festgelegt ist, in der die einzelnen Templates angewendet werden. So muß z.B. *X wa Y* vor *X no Y* angewendet werden. Allerdings ist dies nur eine Halbordnung, so daß nicht immer klar ist, welches Template als nächstes angewendet werden muß. Somit können auch mehrere Parsebäume entstehen, was bei längeren Sätzen zu Performanceproblemen führen kann. In [FI96] ist eine Möglichkeit beschrieben, wie mit Hilfe der Beispielsätze unwahrscheinlichere Bäume gleich herausgefiltert werden können und damit Zeit eingespart werden kann.

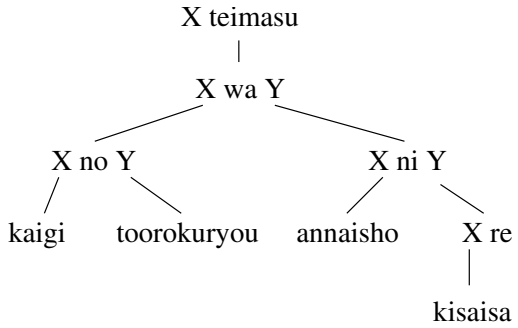

Abbildung 2.1: Der Syntaxbaum des japanischen Beispielsatzes

Da das System auch vom Englischen ins Japanische übersetzt, ist eine englische Grammatik vorhanden, die nach dem gleichen Prinzip funktioniert. Allerdings besitzt die englische Sprache nicht so viele Funktionswörter wie die Japanische. Um einen Satz sinnvoll zerlegen zu können, müssen künstliche Funktionswörter eingefügt werden. So wird z.B. in *The bus leaves Kyoto at eleven* noch die künstlichen Funktionswörter *noun-verb* und *verb-propn* eingefügt. Der Satz lautet nun:

The bus noun-verb leaves verb-propn Kyoto at eleven

Hier ergeben sich zwei mögliche Parsebäume, da die Templates *X verb-propn Y* und *X at Y* nicht geordnet sind. Die entstandenen Parsebäume sind in Abb. 2.2 dargestellt.

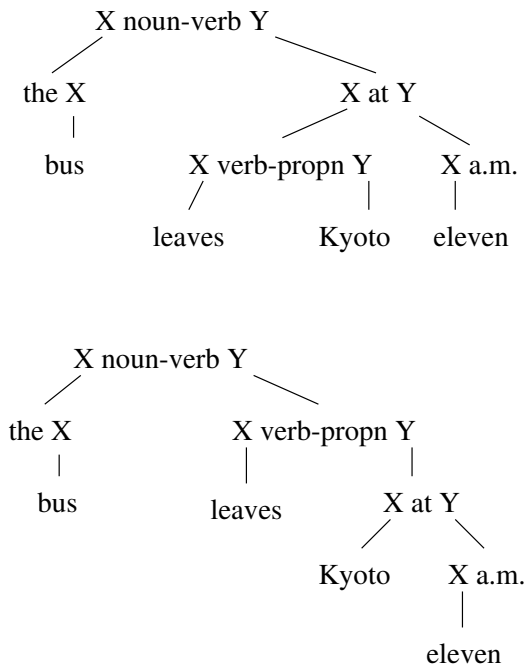

Abbildung 2.2: Englischer Satz mit zwei verschiedenen Parsebäumen

Auswahl der Translation Templates

Für jedes Template in der Ausgangssprache muß jetzt ein passendes Template in der Zielsprache gefunden werden. Dabei bestehen meist mehrere Alternativen, so hat z.B. das japanische Template *X no Y* als englische Entsprechungen u.a. *Y of X*, *Y for X* und *Y in X*. Um das zum Kontext passende Template zu finden, sind jedem englischen Template eine Reihe japanischer Wortpaare zugeordnet, wie in Tab. 2.1 zu sehen ist. Das erste Wort wird dabei mit *X* verglichen, das zweite mit *Y*. Der Wortvergleich erfolgt durch eine Distanzfunktion, die ermittelt, wie semantisch ähnlich die beiden Wörter sind. Anschließend werden beide Werte noch gewichtet und dann zusammengerechnet.

X no Y	Y of X	((ronbun{paper}, daimoku{title}), ...)
	Y for X	((hoteru{hotel},yoyaku{reservation}), ...)
	Y in X	((Kyooto{Kyoto},kaigi{conference}), ...)

Tabelle 2.1: Verschiedene Übersetzungen von *X no Y*

Bei Templates, die nur Wörter direkt übersetzen und keine Variable enthalten, werden die Beispielwörter mit dem ganzen Satz verglichen. So wird z.B. bei Verwendung der Templates, die in Tab. 2.2 dargestellt sind, *sochira* in dem Satz *sochira ni tutaeru* mit *you* übersetzt, da *tutaeru*{*convey*} mit *okuru* {*send*} ver-

sochira	this	((desu{be} ...))
	you	((okuru{send} ...))
	it	((miru{see} ...))

Tabelle 2.2: Übersetzungen von *sochira*

wandt ist.

Die Funktion, die die Distanz zwischen zwei Wörtern ermittelt, baut auf manuell definierten semantischen Ähnlichkeiten auf, die in einem Thesaurus gespeichert sind. So ist z.B. definiert, daß *yokousumo*{*procedings*} *ronboun*{*paper*} ähnlicher ist, als *hoteru*{*hotel*}. Die einzelnen Ähnlichkeiten zwischen den Wörtern sind konkret so realisiert, daß jedes Wort einer Kategorie zugeordnet ist. So fällt z.B. *ronboun*{*paper*} in die Kategorie *writing* und *yokoushuu*{*proceedings*} in die Kategorie *book*. Die Kategorien selbst sind in einer Baumstruktur angeordnet. So haben z.B. die Kategorien *writing* und *book* als gemeinsame übergeordnete Kategorie *document* (siehe auch Abb. 2.3). Der Wert der Distanzfunktion steigt dabei linear mit der Höhe der ersten gemeinsamen Kategorie im Baum. In diesem Beispiel ist die Distanz 1/3. Dieser Wert ergibt sich dadurch, daß im Baum eine Stufe hochgegangen werden muß, der Baum die Höhe 4 hat und die Distanz zwischen 0 und 1 liegen soll.

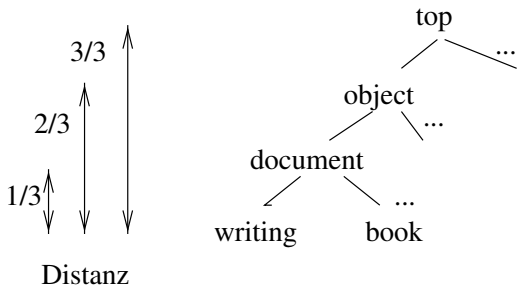

Abbildung 2.3: Attributsbaum

Nachdem die Distanz zwischen den Wörtern ermittelt ist, müssen die Einzelwerte noch zu einem gemeinsamen Wert gemittelt werden. Dies ist natürlich nur nötig, wenn mehrere Variable im Template vorkommen. Anstatt einfach den Mittelwert zu nehmen, wird berücksichtigt, wie relevant jedes Wort für die Auswahl des Templates ist. So ist z.B. bei dem Template-Paar

$$X\ no\ Y \rightarrow X\ Y\ ((mittsu\{three\},hoteru\{hotel\}),...)$$

viel wichtiger, daß X eine Zahl ist als daß Y Ähnlichkeit mit einem Hotel hat. Der Grund liegt darin, daß im Englischen Zahlen immer direkt vor dem Nomen stehen, unabhänigig davon, zu welcher semantischen Kategorie das Nomen gehört.

Generierung

Nachdem für alle ausgangssprachlichen Templates eine Übersetzung ausgewählt wurde, werden diese Übersetzungen in den Syntaxbaum eingesetzt. Der Baum besteht somit nur noch aus zielsprachlichen Templates (siehe Abb. 2.4). Zur Generierung des englischen Satzes wird der Baum nun *top-down* durchlaufen. Die Wörter können dabei aber nicht einfach in *infix-order* ausgegeben werden, da im Baum Marker wie z.B. *sub* auftauchen können. Diese Marker beinhalten syntaktische Informationen, in diesem Fall, daß es sich bei dem linken Teilbaum um ein Subjekt handelt. Ein anderer Marker ist *obj*, der den linken Teilbaum als Objekt deklariert und Verwendung findet, wenn Subjekt und Objekt vertauscht werden sollen. In diesem Fall weicht die Ausgabe der Wörter recht stark von der Reihenfolge der Wörter im Baum ab.

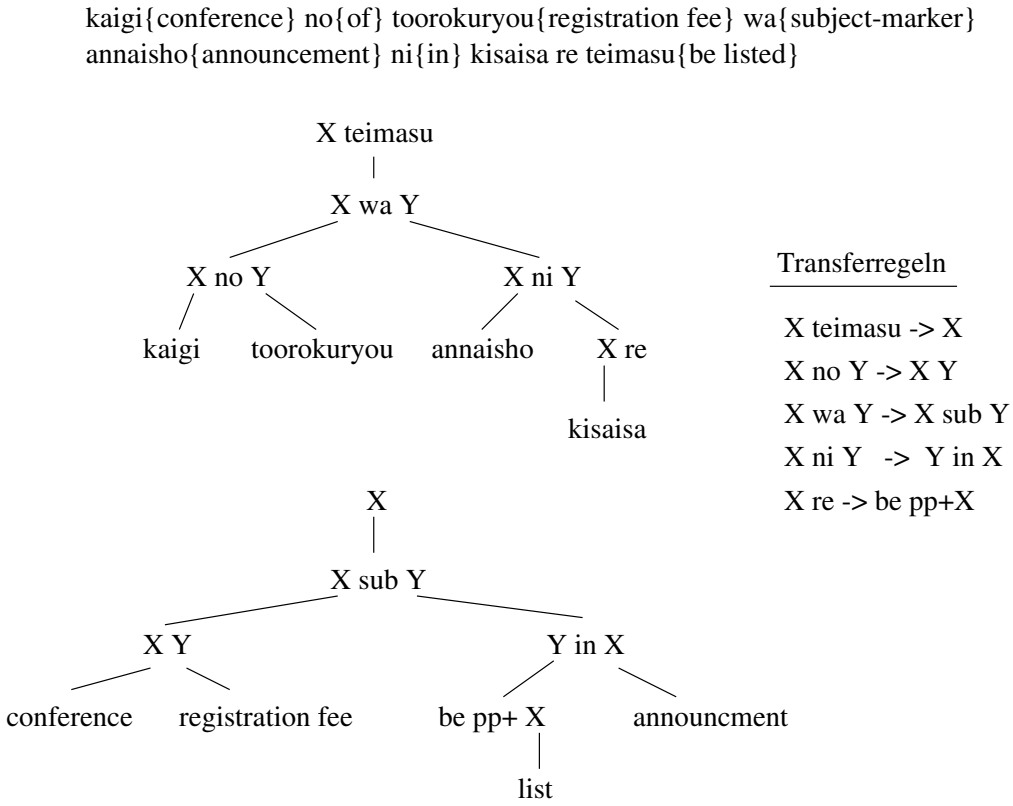

Abbildung 2.4: Der generierte englische Syntaxbaum. Zur Veranschaulichung der Arbeitsweise ist noch der japanische Syntaxbaum zusammen mit den verwendeten Template-Paaren abgebildet

2.2 Generierung von Translation Templates

Translation Templates, die z.B. im System vom ATR benötigt werden, können entweder manuell erstellt oder automatisch aus bilingualen Korpora extrahiert werden. Ein Verfahren, um Templates automatisch zu generieren, wird von H. Kaji, Y. Kida und Y. Morimoto in [KKM92] vorgeschlagen, die am Hitachi-Forschungslabor in Kawasaki arbeiten. Das Verfahren ist dabei für ein japanisch-englisches Übersetzungssystem beschrieben.

Die Templates bestehen aus einem Satz oder Satzteil, wobei einzelne Wörter bzw. Phrasen durch Variable ersetzt werden – ganz ähnlich wie bei dem ATR-System. Damit der Algorithmus eingesetzt werden kann, müssen in dem bilingualen Text bereits einzelne Sätze miteinander gekoppelt sein, d.h. jedem Satz in der Ausgangssprache muß ein Satz in der Zielsprache entsprechen. Außerdem wird ein zweisprachiges Wörterbuch benötigt.

Der vorgeschlagene Algorithmus, der aus einem Satzpaar Templates generiert, besteht aus folgenden Schritten:

- Syntaktische Analyse der Sätze in der Ausgangssprache und in der Zielsprache.
- Anhand des Wörterbuchs wird versucht, Wörter in beiden Sätzen zu koppeln.
- Mit Hilfe der gekoppelten Wörter wird versucht, Phrasen (wie NP) zu koppeln.
- Gekoppelte Wörter und Phrasen werden z.T. durch Variable ersetzt und damit Templates generiert.

Die generierten Templates werden am Ende nochmal überarbeitet. Sich widersprechende Templates werden aussortiert oder die Variablen werden mit Bedingungen versehen.

Syntaktische Analyse

Die syntaktische Analyse des Ausgangs- und Zielsprachensatzes erfolgt anhand einer beliebigen Grammatik. Bei den im Artikel verwendeten Beispielen wurde eine flache Grammatik verwendet, so werden z.B. NPs nicht weiter zerlegt. Wenn mehrere verschiedene Syntaxbäume entstehen, was bei den meisten Grammatiken der Fall ist, kann ein Teil von ihnen beim Koppeln von Phrasen wieder verworfen werden, da Ambiguitäten, die im Englischen vorhanden sind, nicht unbedingt im Japanischen vorkommen müssen und umgekehrt.

Um nun einzelne japanische Wörter mit entsprechenden englischen Wörtern zu koppeln, wird im Wörterbuch nach sämtlichen englischen Übersetzungen für ein japanisches Wort gesucht. Falls eine dieser Übersetzungen im englischen Satz

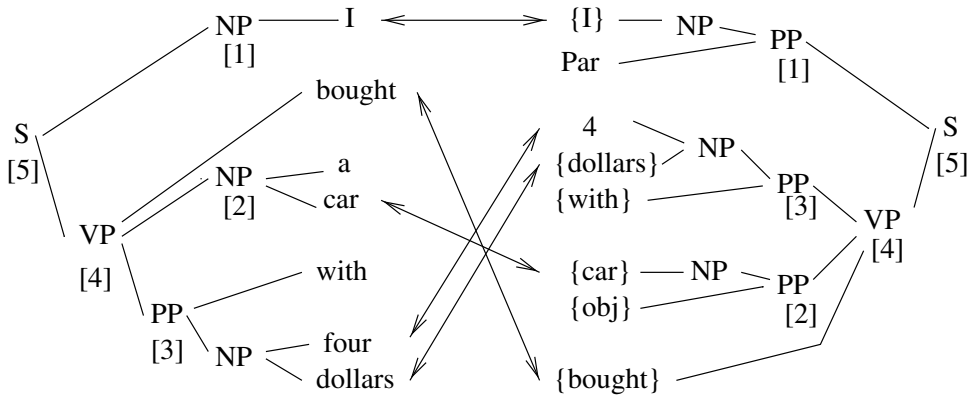

Abbildung 2.5: Satzpaar mit Wort- und Phrasenkopplungen (Phrasenkopplungen sind durch eckige Klammern dargestellt, japanische Wörter sind aus Platzgründen nur durch ihre englischen Entsprechungen angegeben)

gefunden wird, wird das japanische Wort mit dem entsprechenden englischen Wort gekoppelt (s.a. Abb. 2.5).

Werden allerdings mehrere passende Wörter im englischen Satz gefunden, ist keine eindeutige Zuordnung mehr möglich. Deshalb findet vorerst keine Kopplung statt. Erst wenn die Phrasenkopplung abgeschlossen ist, ist eventuell eine Zuordnung innerhalb gekoppelter Phrasen möglich (s.a. Abb. 2.6).

Da gekoppelte Wörter in den Templates durch Variable ersetzt werden können, wird lediglich versucht, Inhaltswörter zu koppeln, aber keine Funktionswörter. Templates, in denen Funktionswörter durch Variable ersetzt sind, wären nicht mehr sinnvoll, da beim Transfer die Variablen mit den entsprechenden Übersetzungen gebunden werden, aber Funktionswörter je nach Kontext sehr unterschiedlich übersetzt werden. Zudem müssen in den Templates noch einige Wörter als Gerüst erhalten bleiben, da Templates, die nur Variable enthalten, keinen Sinn mehr machen.

Koppeln von Phrasen

Nachdem einzelne Wörter gekoppelt wurden, wird nun versucht, auf dieser Basis auch Phrasen miteinander zu koppeln. Für jede japanische Phrase wird nach einer passenden englischen Phrase gesucht. Dabei ist zu beachten, daß Wörter innerhalb der japanischen Phrase nur mit den englischen Wörtern gekoppelt sein dürfen, die innerhalb der englischen Phrase liegen.

Ein Problem entsteht, wenn nicht alle Inhaltswörter gekoppelt werden können. So können mehrere Phrasen durchaus die gleichen gekoppelten Wörter enthalten (s. Abb. 2.7). Der bei diesem System vorgeschlagene Lösungsansatz ist, die jeweils kürzesten und längsten Phrasen zu koppeln.

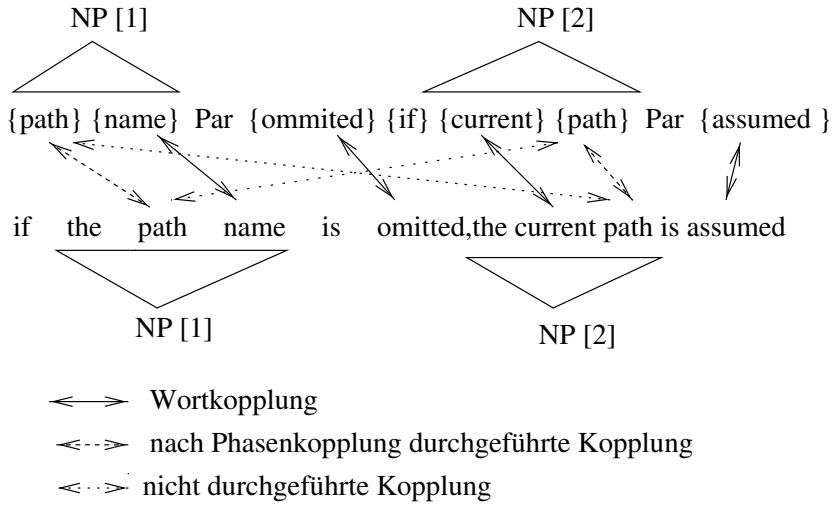

Abbildung 2.6: Satzpaar mit nachträglich gekoppeltem Wort (*path*)

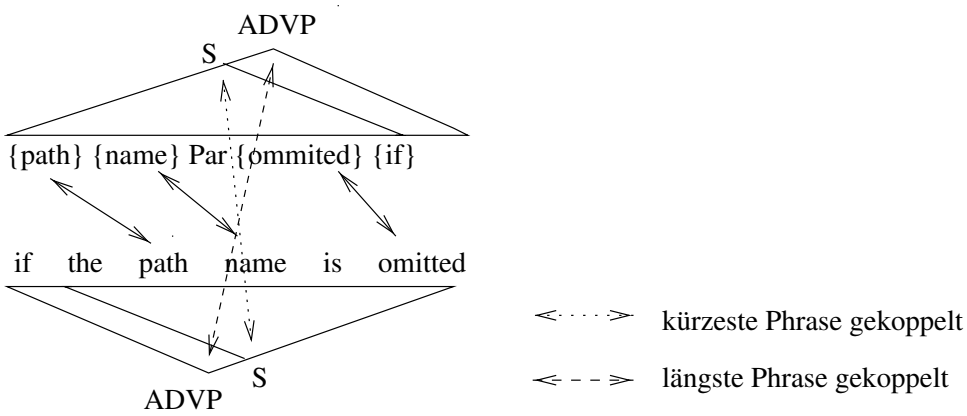

Abbildung 2.7: Kopplung von mehreren Phrasen mit gleichen Wortkopplungen

Generierung der Translation Templates

Die Generierung der Templates aus den Beispielsätzen erfolgt, indem gekoppelte Wörter bzw. Phrasen durch Variable ersetzt werden. Es müssen allerdings nicht alle gekoppelten Wörter bzw. Phrasen durch Variable ersetzt werden. So entstehen meist aus einem Satz mehrere Translation Templates, je nachdem wo Variable eingesetzt werden.

Templates müssen dabei nicht nur aus vollständigen Sätzen bestehen, sondern können auch nur aus Phrasen bestehen. Ein solches Template wird generiert, indem gekoppelte Phrasen das Grundgerüst bilden und die darin enthaltenen

gekoppelten Wörter z.T. durch Variable ersetzt werden.

Überarbeitung der Translation Templates

In einem letzten Schritt werden die aus einem bilingualen Text generierten Templates noch miteinander verglichen. Bei einigen Templates ist der ausgangssprachliche Teil identisch, der zielsprachliche Teil dagegen unterschiedlich. Somit bestehen mehrere Übersetzungsmöglichkeiten. In vielen Fällen hängt die Übersetzung direkt davon ab, woran die Variablen gebunden sind. So hat z.B. *play X[NP]* verschiedene Übersetzungen im Japanischen, je nachdem ob *X* eine Sportart oder ein Instrument ist. Die Variablen können deshalb noch mit semantischen Bedingungen verknüpft werden, z.B. *play X[NP/sport]*.

Selten auftretende Templates werden einfach herausgefiltert, falls häufig auftretende Templates mit gleichem ausgangssprachlichen Teil, jedoch unterschiedlichem zielsprachlichen Teil vorhanden sind. Damit soll verhindert werden, daß sich zuviele Templates ansammeln, deren Übersetzung nur in wenigen Ausnahmefällen korrekt ist.

Ein Problem beim Einsatz von Translation Templates sei hier noch angesprochen: Die Flektionen der Wörter müssen beim Übersetzen noch entsprechend angepaßt werden, wozu noch eine morphologische Komponente in der Zielsprache gebraucht wird.

2.3 Pangloss

Das Pangloss Übersetzungssystem wird gemeinsam von drei Instituten, dem Computing Research Laboratory an der New Mexico State University, dem Information Sciences Institute an der University of Southern California und dem Center for Machine Translation an der Carnegie Mellon Univerity seit 1992 entwickelt ([NBD94], [Nir95], [Bro96]). Es übersetzt vom Spanischen ins Englische und umgekehrt. Drei verschiedene Ansätze werden dabei verfolgt:

- Wissensbasierter Ansatz
- Auf Beispielsätzen basierender Ansatz
- Ein auf lexikalischem Transfer beruhender Ansatz

Bei der Übersetzung laufen die Systeme parallel. Jedes System liefert als Ergebnis nicht einen vollständigen Satz zurück, sondern lediglich Übersetzungen von Satzteilen. Die Systeme versuchen dabei selbst, die Qualität der einzelnen Teilübersetzungen einzuschätzen und zu bewerten. Diese Teilübersetzungen werden anschließend mit Hilfe einer statistischen Komponente zu einer Gesamtübersetzung des Satzes zusammengefügt. Im Rahmen meiner Diplomarbeit ist nur der beispielbasierte Ansatz von Interesse und soll hier vorgestellt werden.

Bei diesem System findet im Gegensatz zum ATR-System keinerlei syntaktische Verarbeitung statt. Verwendung findet lediglich ein Korpus aus Satzpaaren, ein bilinguales Wörterbuch und ein Synonymwörterbuch. Damit werden dann jeweils Teilstücke des Ausgangssatzes übersetzt. Optional können auch noch einige Zusatzinformationen angegeben werden, z.B. Wörter, die durch Wortklassen ersetzt werden sollen, wie Zahlen oder für die Übersetzung weniger wichtige Wörter wie Artikel. Im System wird z.Zt. ein 270 MB großer Korpus (Spanisch-Englisch) eingesetzt.

Die prinzipielle Vorgehensweise ist dabei wie folgt:

- Der Ausgangssatz wird mit Hilfe eines speziellen Algorithmus, der noch vorgestellt wird, in Teilstrings zerlegt. Dabei können unterschiedliche Zerlegungen des Ausgangssatzes entstehen, die alle weiterverarbeitet werden.
- Für jeden dieser Teilstrings wird der Korpus nach passenden Beispielsatzpaaren durchsucht.
- Mögliche Übersetzungen der Teilstrings werden mit Hilfe des bilingualen Wörterbuchs und des Synonymwörterbuchs aus dem Beispielsatzpaar ermittelt.
- Die einzelnen möglichen Übersetzungen werden anhand von Regeln qualitativ bewertet und an diejenige Komponente weitergereicht, die die Ergebnisse der einzelnen Systeme zu einem Satz zusammenfügt.

Nachdem ein Überblick über das System gegeben wurde, wird nun auf die einzelnen Teile näher eingegangen.

Der Ausgangssatz wird zuerst in Teilstrings zerlegt, die sich auch überlappen können. Alle Teilstrings, die mindestens zwei Wörter umfassen und in denen alle Wörter zumindest in einem gemeinsamen Beispielsatz vorkommen, werden weiterverarbeitet. Technisch ist das so realisiert, daß zu jedem Wort eine Liste mit Verweisen existiert, die angibt, in welchen Beispielsatzpaaren das Wort überhaupt vorkommt. Nun wird für jedes Wort im Teilstring diese Liste geladen und der Durchschnitt aus allen Listen gebildet. Falls der Durchschnitt nicht leer ist, gibt es zumindest ein gemeinsames Beispielsatzpaar. Die verschiedenen Zerlegungen des spanischen Satzanfangs *El Banco de Santander habia sido elegido el lunes por las autoridades monetarias* sind in Tabelle 2.3 beschrieben.

El Banco de	the Bank of
El Banco de Santander	the Bank of Santander
Banco de	Bank of
de Santander	of Santander
habia sido	been
elegido el	chosen the
el lunes por	Monday by the
por las autoridades	by the health authorities
por las autoridades monetarias	by the monetary authorities
las autoridades monetarias	the monetary authorities

Tabelle 2.3: Zerlegung mit Übersetzungen

Für jeden dieser Teilstrings werden nun anhand der Listen mit Verweisen die Beispielsatzpaare tatsächlich in den Speicher geladen. Da normalerweise die Anzahl der zu ladenden Beispielsatzpaare zu groß ist, werden z.Zt. lediglich die letzten fünf Beispielsatzpaare aus der Liste berücksichtigt. So werden neu hinzugefügte Satzpaare bevorzugt.

Das einzige Kriterium für die Auswahl der Beispielsatzpaare ist, daß alle Wörter des zu übersetzenden Teilstrings darin vorkommen. Daher ist es möglich, daß der Teilstring in dem Beispielsatz nicht exakt enthalten ist, sondern die Reihenfolge der Wörter vertauscht ist oder andere Wörter eingefügt sind. Eine Bewertungsfunktion vergibt nun Strafpunkte, je nachdem wie groß die Abweichung zum übersetzenden Teilstring ist. So gibt es z.B. für jede Wortvertauschung 15 Strafpunkte, für jedes zusätzlich eingefügte Wort 5 Strafpunkte. Ein kleines Beispiel zur Verdeutlichung: Ist der zu übersetzende Teilstring $A\ B\ C$ und der quellsprachliche Teil des Beispielsatzpaares $X\ A\ Y\ C\ B\ Z$ ergeben sich $5 + 15 = 20$ Strafpunkte. Die Strafpunkte werden bei der am Ende stattfindenden Auswahl berücksichtigt.

Für den so ermittelten Teilstring muß nun eine entsprechende Übersetzung im zielsprachlichen Teil des Beispielsatzpaares gefunden werden. Dazu wird für jedes Wort im gesamten quellsprachlichen Satz (also nicht nur für den Teilstring) eine Liste mit allen im Wörterbuch aufgeführten Übersetzungen aufgestellt. Diese

werden dann im zielsprachlichen Satz gesucht und falls eine eindeutige Zuordnung möglich ist, können die quell- und zielsprachlichen Wörter gekoppelt werden. Abb. 2.8 zeigt dafür ein Beispiel. *Naciones* kann nicht mit *nations* gekoppelt werden, da *naciones* laut Wörterbuch auch mit *country* übersetzt werden kann.

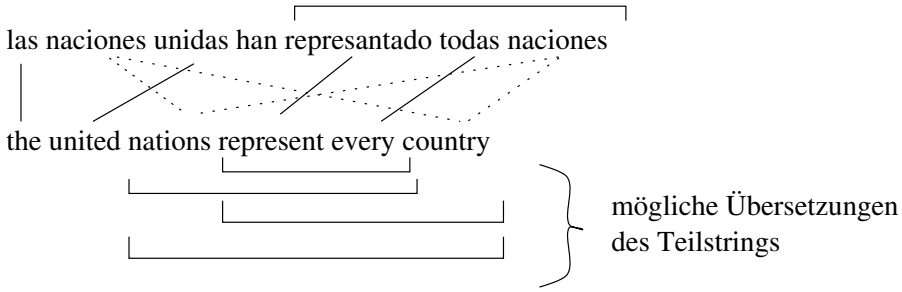

Abbildung 2.8: Kopplung der Wörter in einem Satzpaar mit möglichen Übersetzungen

Alle Teilstrings des zielsprachlichen Satzes, die nur Wörter enthalten, die mit Wörtern des ausgangssprachlichen Satzes gekoppelt sind, kommen als potentielle Übersetzungen in Frage (Beispiel siehe Abb. 2.8). Um nun einen Teilstring auszuwählen, werden die so gewonnen Teilstrings wiederum bewertet und mit den vorher vergebenen Strafpunkten verrechnet. Bei der Bewertung werden dabei u.a. folgende Kriterien berücksichtigt:

- die Anzahl der Wörter, die im Teilstring gekoppelt bzw. nicht gekoppelt sind (in diesem Beispiel 2 gekoppelt und 0-2 nicht gekoppelte Wörter)
- wieviel Wörter im Teilstring haben Entsprechungen, sind aber nicht gekoppelt (z.B. *naciones* und *country*)
- die Übereinstimmung von Satzzeichen bzw. Satzanfang und -ende (z.B. *naciones* und *country* stehen beide am Ende)

Die einzelnen Teilstrings mit allen möglichen Übersetzungsvarianten werden zusammen mit den Bewertungen an eine Komponente weitergegeben. Diese sammelt auch die Übersetzungsvorschläge der anderen Komponenten in einem Graphen und wählt anschließend die günstigste Übersetzung aus.

2.4 STAG-basisierte Übersetzungssysteme

Bei den auf STAGs (Synchronous Tree Adjoining Grammar) basierenden Übersetzungssystemen [ASJ90],[SS90] handelt es sich nicht direkt um beispielbasierte Übersetzungssysteme, sondern um regelbasierte. Wie auch beim ATR-System können diese Regeln automatisch aus Korpora erstellt und somit prinzipiell zu einem beispielbasierten System erweitert werden. Im Gegensatz zum ATR-System wird in den o.g. Artikeln kein Vorschlag gemacht, wie unter mehreren alternativen Regeln eine ausgewählt werden kann, d.h. wie eine Bewertungsfunktion aussehen könnte. Eine prototypische Implementierung findet sich in [Pri97].

Der Übersetzungsformalismus baut auf dem TAG-Grammatikformalismus auf. Die Grundidee bei dem TAG-Formalismus ist, daß der Syntaxbaum aus vorgefertigten Teilbäumen zusammengesetzt wird, die nicht nur aneinandergehängt, sondern auch mitten im Baum eingefügt werden können.

Bei Synchronen TAGs ist jedem vorgefertigten ausgangssprachlichen Teilbaum zusätzlich ein weiterer zielsprachlicher Teilbaum zugeordnet. Wenn nun bei der Analyse des ausgangssprachlichen Satzes der Syntaxbaum aus den einzelnen Teilbäumen zusammmengesetzt wird, entsteht parallel dazu ein zweiter Syntaxbaum aus den entsprechenden zielsprachlichen Teilbäumen. Links zwischen den beiden Teilbäumen markieren dabei die Stelle, an der der zielsprachliche Teilbaum eingefügt werden muß. Das ermöglicht Phrasenverschiebungen, z.B. Vertauschung von Subjekt und Objekt oder die Umwandlung eines Objektes in eine Präpositionalphrase.

Zur Illustration die Übersetzung des englischen Satzes:

Apparently John misses Mary

wird im Spanischen zu

Apparement Mary manque à John

(entnommen aus [ASJ90]). Dabei findet eine Vertauschung des Subjekts und Objekts statt. In Abb. 2.9 sind die einzelnen Teilbaumpaare dargestellt. Abb. 2.10 zeigt ein Zwischenresultat nachdem zwei Teilbäume zusammengesetzt sind. In Abb. 2.11 ist das endgültige Resultat zu sehen.

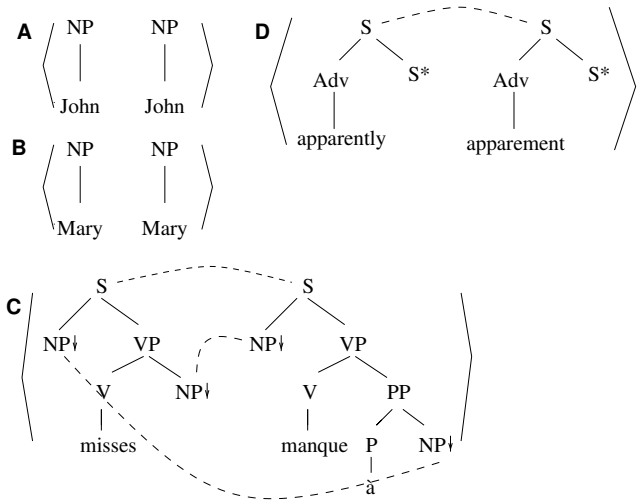

Abbildung 2.9: Beispiel für Baumpaare, die eine STAG bilden

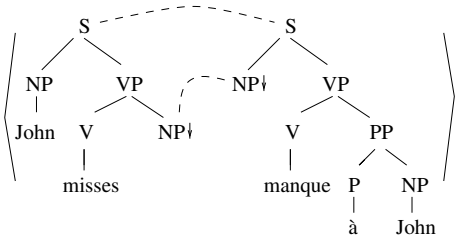

Abbildung 2.10: In den Initialbaum C wurde der Baum für *John* substituiert.

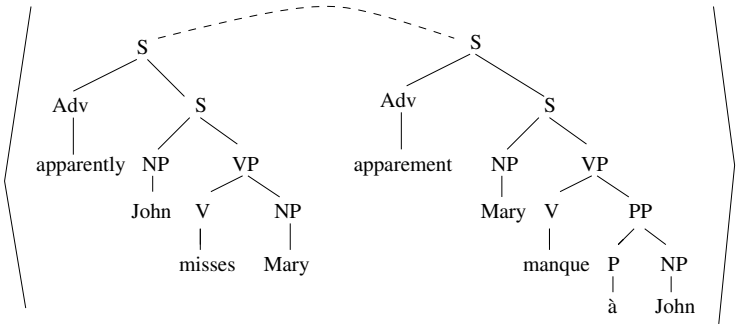

Abbildung 2.11: Zum Schluß noch eine Adjunktion, wobei der Link zwischen den S-Knoten aus dem Baumpaar D stammt.

2.5 Shake-and-bake Übersetzung

Wie bei der STAG-Übersetzung handelt es sich bei der Shake-and-Bake Übersetzung [Bea92],[Whi92] nicht um ein beispielbasiertes System, das aber ebenfalls aufgrund des einfachen Aufbaus zu einem beispielbasierten System erweitert werden kann.

Prinzipiell ist es dem STAG-Verfahren sehr ähnlich, statt eines hierarischen Syntaxbaums findet hier allerdings ein syntaktisches Netzwerk Verwendung. Der Vorteil hiervon ist laut [Whi92], daß das Aussehen des syntaktischen Netzwerks weitgehend unabhängig vom verwendeten Grammatikformalismus ist und somit die gleichen Übersetzungsregeln von unterschiedlichen Grammatiken eingesetzt werden können. Für die Übersetzung existieren wie bei STAG Paare aus Teilnetzwerken, wobei die Knoten, an denen andere Teilnetzwerke andocken können, mit Links verbunden sind (ebenfalls wie bei STAG). Mit Hilfe der Paare und des ausgangssprachlichen Netzes wird ein zielsprachliches Netz aufgebaut. Abschließend muß das entstandene Netzwerk noch linearisiert werden. Dazu wird ein Shift-Reduce-Parser eingesetzt, auf den hier allerdings nicht näher eingegangen werden soll.

Eine Beispielübersetzung des englischen Satzes

Hans likes to swim

in den deutschen Satz

Hans schwimmt gern

ist in den Abb. 2.12 - Abb. 2.13 dargestellt, wobei Abb. 2.12 das ausgangssprachliche und das zielsprachliche Netzwerk zeigt und Abb. 2.13 die Übersetzungspaare. Zu beachten ist, daß in den Übersetzungspaaren Links von Wörtern beeinflußt werden können, die gar nicht mit diesem Paar übersetzt werden, in diesem Beispiel verändert *gerne* beim Übersetzen die Subjektbeziehung des Verbs, dem *gerne* zugeordnet ist.

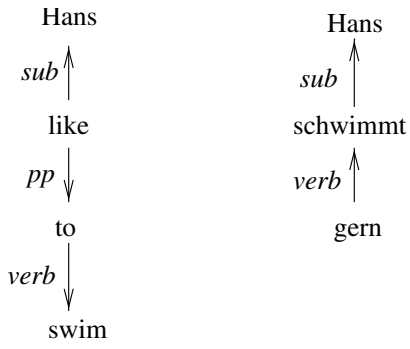

Abbildung 2.12: Syntaktische Beziehungen zwischen den Wörtern bei den Sätzen *Hans schwimmt gern* und *Hans like to swim*

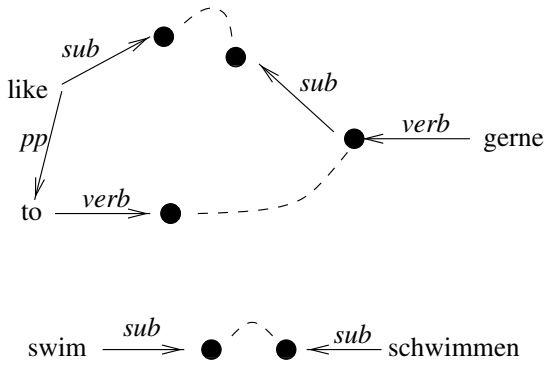

Abbildung 2.13: Übersetzungsregel für das Wortpaar *gerne* und *like to* und *schwimmen*, Links sind gestrichelt gezeichnet

Kapitel 3

System im Überblick

3.1 Ansatz

Mein System basiert, ebenso wie das ATR-System, auf Translation Templates. Diese werden aus einem Beispielsatzkorpus in einem Vorverarbeitungsschritt extrahiert, bei dem der HTG-Vorschlag als Grundlage diente und erweitert wurde. Die Sätze werden dabei ebenfalls wie beim ATR-System und im Gegensatz zum Pangloss-System syntaktisch analysiert.

Meiner Ansicht nach ist eine syntaktische Analyse wünschenswert, da sonst bei der Generierung die Umstellung von Nominal- und Präpositionalphrasen bei der Übersetzung fast unmöglich wird. Auch die Verarbeitung längerer Sätze mit komplizierten Nebensatzstrukturen scheint ohne syntaktische Analyse kaum lösbar. Dies zeigen auch die in [NBD94] und [Bro96] geschilderten Erfahrungen mit der beispielbasierten Komponente des Pangloss-Systems, das ohne Hilfe der anderen Komponenten keine kompletten Sätze übersetzen kann.

Auf eine komplizierte syntaktische Analyse wird bei meinem System bewußt verzichtet. Sie beschränkt sich im wesentlichen auf die Erkennung von Nominal- und Präpositionalphrasen und das Bestimmen von Nebensatzgrenzen. Dadurch kann der Algorithmus für den Baumvergleich, der zum Finden des passendsten Beispielsatzes benötigt wird, und der Alignment-Algorithmus bei der Vorverarbeitung relativ einfach und effizient gehalten werden.

Bei der Transferkomponente habe ich versucht, einige Probleme des ATR-Ansatzes zu vermeiden. So werden im ATR-System die Translation Templates nicht nur zum Übersetzen, sondern auch für die syntaktische Analyse eingesetzt, was eine enge Verzahnung von syntaktischer Analyse und Transferkomponente zur Folge hat. Ein weiteres Problem ist die genaue Entsprechung von ausgangs- und zielsprachlichen Syntaxbaum. So müssen sowohl bei der syntaktischen Analyse (Einfügen künstlicher Funktionswörter im Englischen) als auch bei der Generierung (Berücksichtigung syntaktischer Marker) Kompromisse eingegangen werden.

Um diese Komplikationen zu vermeiden, werden deshalb bei meinem System in einem Schritt nicht einzelne Knoten bzw. Blätter im Syntaxbaum übersetzt. Stattdessen werden mehrere Wörter zusammen übersetzt. Die inneren Knoten, die die

syntaktische Struktur bilden, werden je nach Bedarf sowohl aus dem Ausgangs- als auch dem Beispielsatz übernommen und nicht getrennt übersetzt. Damit kann der ausgangs- und der zielsprachliche Syntaxbaum eine unterschiedliche Struktur besitzen.

Im Gegensatz zum ATR-System wird als Kontext der komplette Beispielsatz herangezogen, aus dem das Translation Template generiert wurde und nicht nur einzelne Wörter. Dieser steht hier im Gegensatz zu den manuell erstellten Templates beim ATR-System ohnehin zur Verfügung.

3.2 Aufbau

Der Übersetzungsvorgang bei meinem System läuft ebenfalls in den drei typischen Phasen Analyse – Transfer – Generierung ab. Zusätzlich ist eine Komponente vorhanden, die die Beispielsatzpaare vorverarbeitet, ehe das Übersetzungssystem zum erstenmal eingesetzt werden kann. Der komplette Aufbau des Systems ist in Abb. 3.1 dargestellt.

Bei der hier verwendeten syntaktischen Analyse handelt es sich um eine relativ flache Analyse: Es werden im wesentlichen Nominal- und Präpositionalphrasen und Nebensatzgrenzen erkannt. Zum Einsatz kommen dabei ein POS-Tagger, der für die einzelnen Wörter die Wortarten ermittelt, und kaskadierte Automaten, die die eigentliche Einteilung vornehmen. Zeitausdrücke werden von einem Tool separat erkannt und auch unabhängig übersetzt.

Die Transferkomponente wandelt die von der syntaktischen Analyse aufgebaute Struktur in eine zielsprachliche Struktur um. Im Gegensatz zum ATR-System wird anstelle manuell erstellter Regeln der vorverarbeitete Beispielsatzkorpus verwendet und ein externes zweisprachiges Wörterbuch hinzugezogen. Die Hauptaufgabe des Transfermoduls besteht darin, für die einzelnen Wörter bzw. für eine Gruppe von Wörtern den jeweils passendsten Beispielsatz zu finden und entsprechende Übersetzungen zu einen zielsprachlichen Syntaxbaum zusammenzubauen.

Der Ablauf ist dabei folgender: Die Wörter des zu übersetzenden Satzes werden in einer Liste gespeichert und diese Liste wird sortiert, um bessere Übersetzungsergebnisse und eine höhere Verarbeitungsgeschwindigkeit zu erreichen (s.a. Kap. 6). Danach wird diese Liste der Reihe nach abgearbeitet, für jedes Wort wird nach der entsprechenden Übersetzung in den Beispielsatzpaaren gesucht, wobei dies unter Berücksichtigung von Kontext geschieht. Zur Zeit wird einfach der komplette zu übersetzende Satz als Kontext genommen. Falls im ausgewählten Beispielsatz das Wort zusammen mit mehreren anderen Wörtern übersetzt wird, werden diese Wörter aus der Liste gestrichen. Außerdem werden nicht nur die einzelnen Wörter, sondern auch die syntaktische Struktur berücksichtigt. Es muß also ein Teilbaum gematcht werden. Die in dem Beispielsatzpaar enthaltenen zielsprachlichen Teilbäume, die die Übersetzung enthalten, werden abschließend zu einem neuen Teilbaum zusammengesetzt. Der Ablauf ist auch in Abb. 3.2 dargestellt.

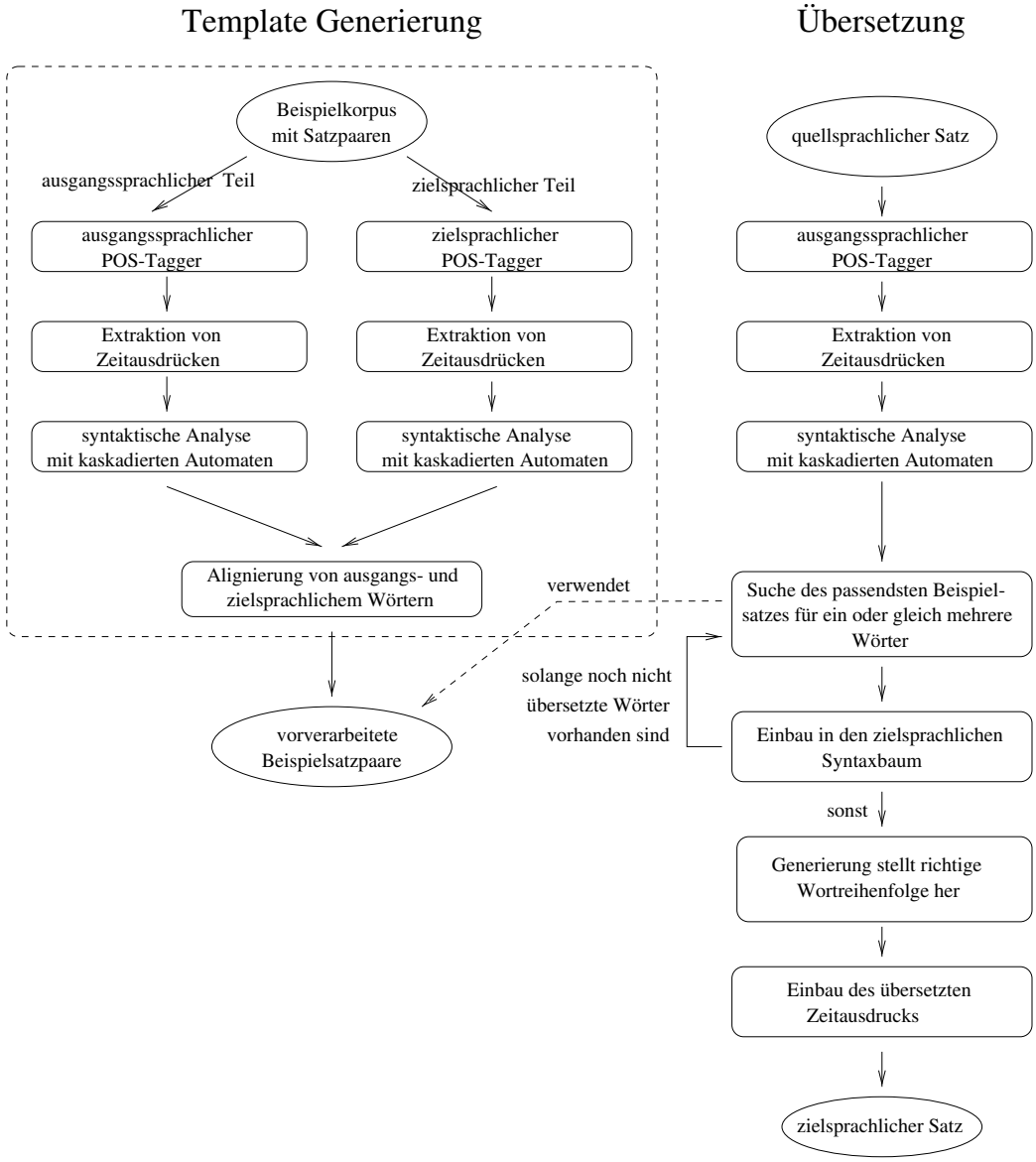

Abbildung 3.1: Der prinzipielle Aufbau des Systems und der Informationsfluß innerhalb des Systems

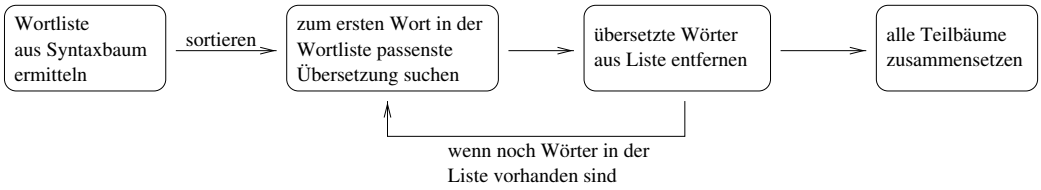

Abbildung 3.2: Der Ablauf innerhalb der Transferkomponente

Die Generierungskomponente sorgt für die richtige Wortstellung bei der Linearisierung des zielsprachlichen Syntaxbaums. Dabei finden, wie bei der Analyse, kaskadierte Automaten Verwendung.

Bei der Vorverarbeitung der Beispielsätze wird versucht, Teile des ausgangssprachlichen und des zielsprachlichen Satzes jeweils als Übersetzungen zuzuordnen. Das Vorgehen entspricht prinzipiell dem HTG-Vorschlag, allerdings wurde das HTG-Verfahren so erweitert, daß die POS der Wörter berücksichtigt und die Satzstruktur stärker einbezogen wird. Dies erlaubt, daß der Umfang des verwendeten Wörterbuchs sehr gering gehalten werden kann.

Folgende Zusammenfassung zeigt, welche Teile des Systems sprachabhängig sind und benötigt werden, wenn das System an neue Sprachpaare angepaßt wird:

- Ein satzweise alignierter bilingualer Korpus
- Ein zweisprachiges Wörterbuch für die Alignmentkomponente, das allerdings nicht besonders umfangreich sein muß
- Ein POS-Tagger, der für Ausgangs- und Zielsprache vorhanden sein muß und die Wortarten für die einzelnen Wörter bestimmt
- Kaskadierte Automaten für die Ausgangs- und Zielsprache, die eine syntaktische Analyse unter Verwendung der POS vornehmen
- Kaskadierte Automaten, die die Generierung in der Zielsprache übernehmen
- Einige Tabellen müssen an die jeweiligen POS der Ausgangs- und Zielsprache angepaßt werden
- Optional kann zusätzlich ein zweisprachiges Wörterbuch zur Wortübersetzung verwendet werden

3.3 Ein Beispiel

Die prinzipielle Funktionsweise des Systems wird in diesem Abschnitt anhand der nachfolgend aufgeführten Beispielsatzpaare und dem Satz

Am Donnerstag habe ich wahrscheinlich keine Zeit

vorgestellt. Auf etliche Details wird an dieser Stelle nicht näher eingegangen. Eine ausführliche Beschreibung findet sich in den entsprechenden Kapiteln.

Folgende Beispielsatzpaare sind vorhanden:

ich habe keine Zeit <T> $\leftrightarrow$ *I am not free <T>*

keine Zeit habe ich da leider $\leftrightarrow$ *unfortunately I have got no time then*

wahrscheinlich $\leftrightarrow$ *probably*

Vorverarbeitung

Bevor die Beispielsatzpaare zur Übersetzung eingesetzt werden, findet eine Vorverarbeitung der Sätze statt. Zuerst erfolgt die Bestimmung der POS für die einzelnen Wörter mit Hilfe eines deutschen und eines englischen POS-Taggers. Danach werden mit kaskadierten Automaten Nominalphrasen (NP), Präpositionalphrasen (PP) und Satzgrenzen ermittelt (siehe Kapitel 4). In einem nächsten Schritt findet das Alignment statt. Dabei werden automatisch die deutschen und englischen Wörter mit Links gekoppelt, um später die entsprechenden Übersetzungen zu finden. Als Hilfe dient dabei ein kleines Wörterbuch, die ermittelten POS und der Syntaxbaum.

Einige Eigenschaften des Alignment-Algorithmus sollen an dieser Stelle noch erwähnt werden:

- Nicht nur einzelne Wörter können zusammengelinkt werden, sondern auch ganze Wortgruppen, die dann zusammen übersetzt werden, wie z.B. *keine Zeit haben*
- Die Satzpaare müssen nicht aus vollständigen Sätzen bestehen, sondern können auch nur einzelne Satzteile oder Wörter enthalten, wie z.B. bei *wahrscheinlich* $\rightarrow$ *probably*
- Links, die von NP-, PP- oder S-Knoten ausgehen, haben noch einen zweiten Link zugeordnet, um bei mehreren gleichen Knoten unterscheiden zu können, welcher ausgangssprachliche Knoten welchem zielsprachlichen Knoten zugeordnet ist. Diese Links werden nachfolgend als *Zweitlinks* bezeichnet.
- Zeitausdrücke werden gesondert behandelt. Sie werden an spezielle PP.T-Knoten und nicht an normale PP-Knoten gekoppelt. In den Beispielsatzpaaren sind die Zeitausdrücke bereits herausgefiltert und durch <T> ersetzt.

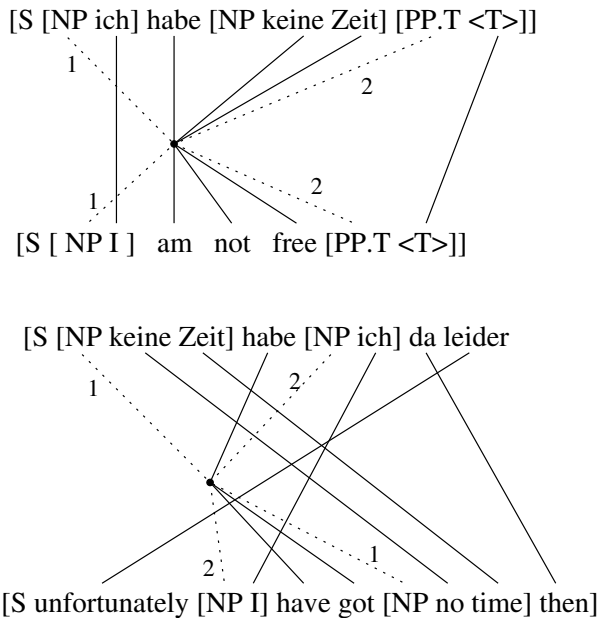

Abbildung 3.3: Die alignierten Beispielsatzpaare. Die gestrichelten Linien gehen zu Knoten, die zusätzlich noch einen Zweitlink haben (hier als Zahl angegeben, siehe Text). Bei mehreren zusammengekoppelten Wörtern sind die Linien durch einen Kreis verbunden.

Übersetzung

Nachdem die Satzpaare aligniert sind, kann die eigentliche Übersetzung beginnen. Als erster Schritt werden die Zeitangaben aus dem zu übersetzenden Satz mit einem speziellen Tool extrahiert. Dieses Tool baut ebenfalls auf kaskadierten Automaten auf und liefert einen TEL-Zeitausdruck [End98] zurück. Anschließend wird der Satz wie die Beispielsatzpaare mit einen POS-Tagger und der Automatengrammatik verarbeitet. Das Ergebnis ist nachfolgend dargestellt:

[*S* [*PP.T tempex(dom:thu)*] [*habe VVFIN*] [*NP* [*ich PPER*]]
[*wahrscheinlich ADV*] [*NP* [*keine ADJA*] [*Zeit NN*]]

Der Satz wird nun schrittweise von links nach rechts übersetzt.[1] Der TEL-Ausdruck wird vorerst unverändert übernommen und erst bei der Generierung ins Englische transferiert. Dabei werden der S- und der PP.T-Knoten von dem Transfermodul automatisch mitkopiert. Der englische Satz sieht dann so aus:

[*S* [*PP.T tempex(dom:thu)*]]

[1] In dem System ist die Vorgehensweise tatsächlich etwas anders, eine verständliche Beschreibung würde aber an dieser Stelle den Rahmen sprengen und vom wesentlichen ablenken, daher sei auf Kap. 6 verwiesen.

Als nächstes wird für *habe* eine Übersetzung gesucht. In den Beispielsatzpaaren ist in den ersten beiden Sätzen *habe* enthalten. Beide Sätze werden mit dem zu übersetzenden Satz verglichen. In diesem Fall schneidet der erste Satz besser ab, da er im Gegensatz zum zweiten einen Zeitausdruck <T> enthält. In Abb. 3.4 sind die Bewertungen für beide Sätze dargestellt. Wie diese Werte zustandekommen, ist ausführlich in Abschnitt 6.2 erläutert. Die Idee dabei ist, daß der zu übersetzende Satz und der aussgangssprachliche Teil des Beispielsatzpaares "übereinandergelegt" werden, d.h. für jedes Wort im Beispielsatz wird ein entsprechendes Wort im zu übersetzenden Satz gesucht. Die Reihenfolge der Wörter spielt dabei im übrigen keine Rolle, es findet eine rein mengenmäßige Betrachtung statt. Je nachdem wie gut die Wörter übereinstimmen, erfolgt die Bewertung. Außerdem werden die einzelnen Wörter noch nach ihrer Wichtigkeit mittels in einer Tabelle gespeicherten Gewichte bewertet, Adverbien werden z.B. in diesem Fall als weniger wichtig eingestuft. Die gestrichelten bzw. kursiv dargestellten Teile sind nur im zu übersetzenden Satz vorhanden, d.h. haben keine Entsprechung im Beispielsatz und können deshalb zu negativen Bewertungen führen.

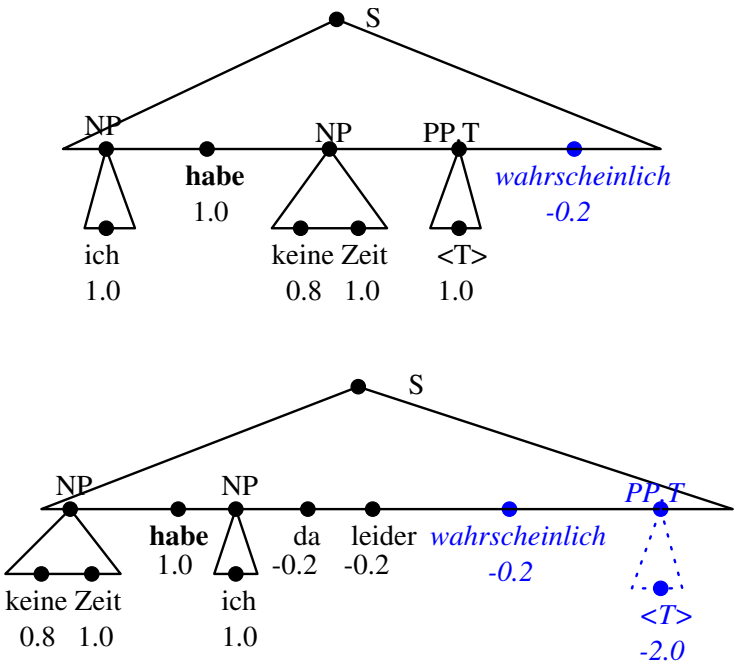

Abbildung 3.4: Bewertung der beiden Beispielsätze bei der Übersetzung von *haben*

Da *habe* zusammen mit *keine Zeit* gekoppelt ist, wird *keine Zeit* gleich mitübersetzt. Außerdem wird der NP-Knoten gleich mitkopiert, da er auch mit den Wörtern gekoppelt ist. Der ebenfalls gekoppelte PP.T-Knoten existiert bereits. Damit die Generierung vereinfacht wird, wandert der PP.T-Knoten noch an die Stelle, in der er auch im zielsprachlichen Teil des Beispielsatzpaares steht. Der im Aufbau befindliche englische Satz hat dann folgendes Aussehen:

[*S* [*NP*] *am not free* [*PP.T tempex(dom:thu)*]]

Als nächstes wird *ich* übersetzt. Für die Übersetzung wird wieder der erste Beispielsatz herangezogen. *I* wird in den bereits angelegten NP-Knoten kopiert. Daraus resultiert

[*S* [*NP I*] *am not free* [*PP.T tempex(dom:thu)*]]

Für die Übersetzung von *wahrscheinlich* ist lediglich das Wort *probably* vorhanden ohne zusätzlichen Kontext. Deshalb kann keine Bewertung vorgenommen werden. Die Übersetzung *probably* wird einfach ans Ende des sich aufbauenden englischen Satzes kopiert. Es ist dann die Aufgabe der Generierungskomponente *probably* richtig einzuordnen. Der englische Satz sieht demnach folgendermaßen aus:

[*S* [*NP I*] *am not free* [*PP.T tempex(dom:thu)*] *probably*]

Keine Zeit wurde ja bereits übersetzt und kann übersprungen werden.

Die Generierungskomponente, die ebenfalls kaskadierte Automaten einsetzt und die Wörter innerhalb eines Knotens in die richtige Reihenfolge bringt, hat in diesem Fall nicht viel zu tun. Nur *probably* muß an die richtige Stelle gerückt und der TEL-Ausdruck ins Englische übersetzt werden.

Das endgültige Resultat sieht dann folgendermaßen aus:

[*S* [*NP I*] *am probably not free on* [*PP.T Thursday*]]

Durch einfache Linearisierung von links nach rechts entsteht dann der zielsprachliche Satz.

3.4 Unterschiede zum HTG- und ATR-System

Da die Kombination aus HTG- und ATR-System vom Prinzip her dem hier beschrieben System ähnelt, sind die wesentlichen Unterschiede zwischen beiden Systemen nochmals aufgelistet:

- Die Knoten werden von der Transferkomponente nicht eins zu eins übersetzt, d.h. einem Knoten in dem ausgangssprachlichen Syntaxbaum entspricht genau ein Knoten im zielsprachlichen Syntaxbaum, sondern beliebige Teilbäume des ausgangssprachlichen Syntaxbaums können zu beliebigen zielsprachlichen Teilbäumen übersetzt werden. Die einzelnen Teilbäume dürfen sich beim Übersetzen sogar überlappen, solange die Blätter exklusiv einem Teilbaum zugeordnet sind. Dadurch ist eine flexiblere Übersetzung möglich, da die Übersetzungsregeln nicht auf einen Knoten beschränkt bleiben müssen.

- Für die Bewertungsfunktion, die zur Auswahl des passendsten Beispielsatzes genommen wird, wird der komplette Satz herangezogen und nicht nur einzelne Wörter. Außerdem existiert für alle zu übersetzenden Wörter ein gleiches Bewertungsschema und ist nicht unterschiedlich, je nachdem ob das Wort ein Knoten oder ein Blatt im Syntaxbaum ist.

- Anstelle von Translation Templates mit Variablen werden Links wie bei dem Shake-and-Bake-Verfahren und beim STAG-Verfahren eingesetzt. Dies liegt im wesentlichen daran, daß bei meinem System im Gegensatz zum ATR-System der komplette Beispielsatz als Kontext zur Verfügung stehen soll. Dieser müßte daher ohnehin jedem Translation Template mitgegeben werden. Daher sind in diesem Fall Links wesentlich kompakter, stellen aber funktional keinen Unterschied dar.

- Zeitangaben werden von einem separaten Tool übersetzt.

Kapitel 4

Syntaktische Analyse

4.1 Einführung

Ziel und Aufgabe der syntaktischen Analyse ist die Erkennung von Nominalphrasen (NPs) und Präpositionalphrasen (PPs) und die Abgrenzung von Nebensätzen.

NPs und PPs zu erkennen, ist bei diesem Ansatz wichtig, weil es sich um austauschbare Einheiten handelt. Bei der Verwendung von Beispielsatzpaaren können sich NPs und PPs ändern, die durch die korrekte Übersetzung ersetzt werden müssen. Um diese Ersetzung durchführen zu können, müssen die Grenzen der NPs und PPs bekannt sein.

Das zweite Ziel ist die richtige Erkennung von Nebensatzgrenzen. Dies ist sehr wichtig, da auf Knoten-Ebene eine mengenorientierte Verarbeitung stattfindet. Wenn nun zwei Vollverben innerhalb eines Knotens auftreten, kann die Transferkomponente die NPs, PPs und die restlichen Wörter wie Adverbien nicht mehr eindeutig den einzelnen Verben zuordnen.

Für diese Aufgabe sind normale Grammatiken wie HPSG oder TAG in der Implementierung zu aufwendig. Hier bieten sich Automaten an, da sie relativ robust und schnell arbeiten und nicht sehr aufwendig zu implementieren sind. Automaten werden deshalb auch vielfach im Information Retrival angewandt, z.B. [Fas98], und zur einfachen syntaktischen Analyse eingesetzt. Beispiele dafür sind die Erkennung von Nominalphrasen [Sen98], die näherungsweise Umsetzung von Phrasenstrukturgrammatiken in endliche Automaten, um schneller parsen zu können [Eva97] und der Einsatz von endlichen Automaten, um die Spracherkennung zu verbessern [ZW98]. Umfangreichere syntaktische Analyse wird von Steven Abney durchgeführt [Abn91], [Abn96].

Mit Hilfe eines Tcl/Tk-Tools zum graphischen Editieren können die Automaten komfortabel entworfen und getestet werden. Das Automaten-Programm selbst ist objektorientiert in C++ geschrieben. Eine detaillierte Beschreibung dazu findet sich im Kapitel 7.

Zunächst werden die verwendeten POS-Tagger vorgestellt, dann wird die prinzipielle Funktionsweise der kaskadierten Automaten beschrieben und anschließend

auf die darauf aufbauende Grammatik eingegangen, die im Deutschen 35 Automaten und im Englischen 19 Automaten umfaßt.

4.2 Verwendete POS-Tagger

Für die Erkennung der POS (part of speech) im Deutschen kommt ein Tagger zum Einsatz, der am IMS (Institut für maschinelle Sprachverarbeitung) an der Universität Stuttgart entwickelt wurde [POS98]. Der verwendete Tagger wurde speziell auf VERBMOBIL-Korpora trainiert und arbeitet mit *decision trees*. Zur Bestimmung der englischen POS wird ein regelbasierter Tagger eingesetzt [Tag98], der mit der *Penn Treebank* trainiert wurde.

In den Tabellen 4.1 und 4.2 werden die wichtigsten POS fürs Deutsche und Englische vorgestellt, wobei die POS in Gruppen eingeteilt sind. Eine ausführliche Erläuterung der POS findet sich fürs Deutsche in [STST95], fürs Englische in [San95].

Nomen	NN
Eigennamen	NE
Artikel	ART PPOSAT PROAT PRELAT PIAT
Pronomen	PPER PPERRF POSS PROS PWS PROAV PRF PDS
fintes Verb	VVFIN
infinites Verb	VVINF
finites Hilfsverb	VAFIN
finites Modalverb	VMFIN
zu-Verben	VVIZU VAIZU VMIZU
Adverb	ADV
Adjektiv	ADJA ADJD PIS PIDAT ORD CARD
Präpositionen	APPR APPRART APZR APPO
Konjunktionen	KOUS KON PAV PWAT PWAV KOKOM
abgetr. Verbzusatz	PTKVZ

Tabelle 4.1: Die wichtigsten POS im Deutschen

4.3 Aufbau der kaskadierten Automaten

In diesem Abschnitt wird detailliert auf die bei diesem System gewählte Implementierung der kaskadierten Automaten eingegangen. In Abb. 4.1 ist die nachfolgende Beschreibung an einem Beispiel veranschaulicht.

Kaskadierte Automaten bestehen aus einzelnen Ebenen, die jeweils eine Liste mit Automaten beinhalten. Jede Ebene übernimmt als Eingabe die Ausgabe der nächst höheren Ebene und reicht wiederum diese Ausgabe an die unter ihr liegende Ebene weiter. Die Ein- bzw. Ausgabe besteht dabei aus Token, die entweder

Nomen	NN NNS NNP
Eigennamen	NE
Artikel	DT PRP
Pronomen	PRP WDT WP
Verb	VB VBPP VBP VBZ
Modalverb	MD
Adverb	RB
Adjektiv	JJ JJR JJS CD
Präpositionen	IN
Wort *to*	TO
Konjunktionen	CC WRB

Tabelle 4.2: Die wichtigsten POS im Englischen

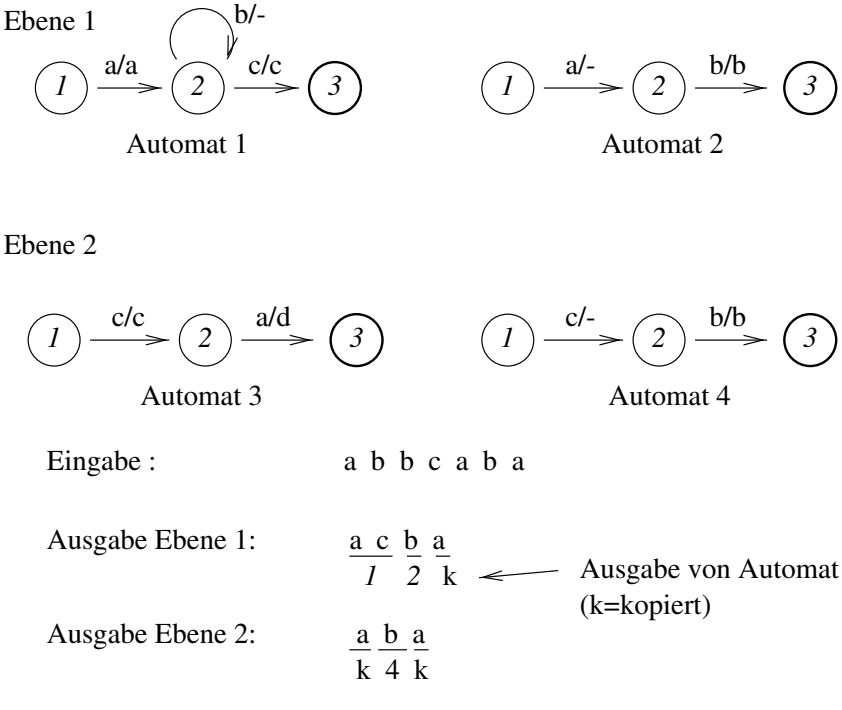

Abbildung 4.1: Beispiel für einen kasadierten Automaten

Wörter mit ihrer POS und Stammform oder komplexere Objekte sind. Diese Objekte enthalten wiederum eine Liste mit Token. NP- und PP-Strukturen werden z.B. als solche Objekte realisiert.

Die Verarbeitung auf einer Ebene läuft nach folgendem Algorithmus ab:

1. Der erste Automat in der Liste versucht, so weit wie möglich den Eingabestrom zu parsen. Falls der Automat erfolglos ist – die genaue Bedeutung wird später erläutert – versucht der nächste Automat, so weit wie möglich

zu parsen. Falls auch dieser Automat keinen Erfolg hat, wird weiter mit dem nächsten probiert etc. Die Liste mit den Automaten wird solange durchlaufen, bis ein Automat Erfolg hat. Kann kein Automat aus der Liste ein Token verarbeiten, wird anstelle der Ausgabe des erfolgreichen Automaten einfach das nächste Token vom Eingabestrom direkt in den Ausgabestrom kopiert.

2. Falls der Eingabestrom in Schritt 1 noch nicht bis zum Ende geparst wurde, wird erneut Schritt 1 mit dem ersten Automaten aus der Liste und dem nächsten Token ausgeführt, das noch nicht bearbeitet wurde. Wurde der Eingabestrom vollständig geparst, wird die nächste Automatenebene aufgerufen.

Das ganze noch als Pseudo-Code:

```
( 1)  foreach level l in automatLevels do
( 2)    while input.isNotEmpty() do
( 3)      sucess = false
( 4)      foreach automat a in l.automatons do
( 5)        if not sucess then
( 6)          sucess = a.parse(input,output)
( 7)        end
( 8)      end
( 9)      if not sucess then
(10)        output.add(input.next())
(11)      end
(12)    end
(13)  end
```

Die einzelnen Automaten selbst werden deterministisch abgearbeitet. Jeder Übergang zwischen zwei Zuständen besteht aus einer Bedingung und einer Aktion (im Automaten als *Bedingung/Aktion* geschrieben). In einem ersten Durchlauf wird mit Hilfe der Bedingungen der Eingabestrom so weit wie möglich geparst, ohne etwas auf den Ausgabestrom zu schreiben. Falls dieser erste Durchlauf Erfolg hatte, d.h. ein Endzustand erreicht werden konnte, werden in einem zweiten Durchlauf die entsprechenden Aktionen ausgeführt. Die Aktionen erst in einem zweiten Durchlauf auszuführen hat den Vorteil, daß keine Undofunktionen für Aktionen nötig sind, die globale Änderungen vornehmen.

Bedingungen testen normalerweise, ob ein bestimmtes Token auf dem Eingabestrom steht. Es können aber auch definierte Funktionen aufgerufen werden, die komplexere Sachverhalte abprüfen (z.B. ob ein Token irgendwann im Eingabestrom vorkommt). Die Funktionen können auch Argumente haben. Die eingebauten Funktionen werden zusammen mit der syntaktischen Analyse im nächsten Abschnitt vorgestellt. In Tabelle 4.3 sind alle implementierten Funktionen aufgelistet.

Bedingungen können zusätzlich mit Prioritäten versehen werden, um eine Reihenfolge festzulegen, in der die Bedingungen abgearbeitet werden. Die Prioritäten sind dabei als Zahlen kodiert: je kleiner die Zahl, desto größer die Priorität.

Aktionen schreiben normalerweise ein Token auf den Ausgabestrom. Aktionen können aber ebenfalls eingebaute Funktionen aufrufen, die komplexere Aufgaben ausführen.

Der Algorithmus zum Durchlaufen eines Automaten sieht als Pseudo-Code folgendermaßen aus:

```
( 1)  method parse(input,output)
( 2)    path = []
( 3)    sucess = true
( 4)    while sucess do
( 5)      <sucess,action> = doTest(input)
( 6)      if (sucess)
( 7)        path.add(action)
( 8)      end
( 9)    end
(10)    path.cutAfterLastFinalState()
(11)    if path.length()==0 then
(12)      return false
(13)    end
(14)    foreach action a in path do
(15)      a.doAction(output)
(16)    end
(17)    return true
(18)  end
```

wobei `doTest`(5) die verschiedenen Übergänge testet, die einen Zustand verlassen. Bei Erfolg wird der entsprechende Zustandswechsel ausgeführt. Außerdem liefert die Funktion die am Übergang stehende Aktion zurück. Die Methode `cutAfterLastFinalState`(10) stellt sicher, daß nur Aktionen bis zum letzen durchlaufenen Endzustand ausgeführt werden. `doAction`(15) führt die Aktion aus.

4.4 Aufbau der Grammatik

Die syntaktische Analyse läuft sowohl im Deutschen als auch im Englischen in folgenden Schritten ab:

- Extraktion von Zeitausdrücken
- Erkennung von NPs und PPs

Bedingungen

`*` immer wahr (Priorität 100)

`-` liest kein Eingabezeichen (Priorität 110)

`scan(<Bedingung>[,back])` wahr wenn irgendwo auf dem Eingabestrom die Bedingung erfüllt ist, geht dabei kein Zeichen auf dem Eingabestrom vorwärts. Wenn back angegeben ist, wird rückwärts auf dem Eingabestrom gesucht.

Kombinierte Bedingungen und Aktionen

`modifyIn(<Chunk-Name>)` ersetzt den Eingabe- und Ausgabestrom durch die im Chunk gespeicherten Token. Dabei werden alle Unterchunks des gerade gelesenen Token mit entsprechendem Namen verarbeitet. Dazu wird der Automatenlevel entsprechend häufig wiederholt, bis alle Chunks abgearbeitet sind

`down(<Chunk-Name>)` ersetzt den Eingabe- und Ausgabestrom durch die im Chunk gespeicherten Token, falls das gelesene Token ein Chunk mit dem Namen ist

`up()` schaltet wieder auf den normalen Eingabe- und Ausgabestrom zurück (Priorität 110)

`back([<Bedingung>])` geht auf dem Eingabestrom eins zurück, wenn <Bedingung> erfüllt ist

`call(<Automaten-Name>)` ruft einen Unterautomaten auf

`subAutomaton()` muß als erste Bedingung in einem Unterautomaten stehen

`return()` kehrt aus einem Unterautomaten zurück

Aktionen

`chunk(<Name>)` legt neuen Chunk an, alle danach auf den Ausgabestrom geschriebenen Token landen im Chunk

`chunkEnd()` beendet einen Chunk, Ausgabestrom ist danach wieder der Ausgabestrom vor dem Aufruf von `chunk`

`delete()` legt das gelese Token nicht auf den Ausgabestrom

`word(<Schreibweise>,<POS>,<Stammform>)` schreibt ein entsprechendes Wort auf den Ausgabestrom

`changePos(<neue-POS>)` ändert im Wort die POS

`verb(fin|pfix)` Fügt Verb und abgetrennten Verbzusatz zusammen

Tabelle 4.3: implementierte Funktionen (Default-Priorität 50)

- Ermittlung von Nebensatzgrenzen
- Zusammenfügen von Verben und abgetrennten Verbzusätzen

Zunächst werden Zeitausdrücke mittels eines Tools extrahiert, das am DFKI in der Dialoggruppe entwickelt wurde. Das Tool verwendet wie die syntaktische Analyse hierarchische Automaten. Dafür wird das gleiche Programm eingesetzt, jedoch um einige zusätzliche Bedingungen und Aktionen ergänzt. Mit dem Tool können sowohl natürlichsprachliche deutsche Zeitausdrücke in TEL-Zeitausdrücke überführt werden, als auch TEL-Zeitausdrücke in englische Sprache umgewandelt werden. Im Prinzip wäre es auch möglich, die deutschen Zeitausdrücke mit dem normalen Übersetzungsmechanismus zu übersetzen, allerdings ist die Abdeckung bei Verwendung dieses Tools viel höher als bei der Verwendung der vorhandenen Beispielsatzpaare.

Die Automaten in der syntaktischen Analyse arbeiten hauptsächlich mit den POS der Wörter, so auch die *NP-Erkennungsautomaten*. Ist eine NP erkannt, wird auf den Ausgabestrom ein Objekt geschrieben, in dem die verarbeiteten Token abgespeichert werden. Das ist so realisiert, daß die Aktion `chunk(NP)` als erste Aktion im Automaten aufgerufen wird. Alle Token, die danach ausgegeben werden, landen nun nicht mehr auf dem normalen Ausgabestrom, sondern werden in dem Objekt in einer Liste abgespeichert. Die nächste Automatenebene erkennt dann nur noch das angelegte Objekt und nicht mehr die darin abgespeicherten Token. Abb. 4.2 zeigt den wichtigsten NP-Erkennungsautomaten fürs Deutsche, der die Kombination *optionaler Artikel, optionales Adjektiv und Nomen* erkennt. Außer diesem existiert noch ein Automat, der die Kombination *Artikel, optionale Adjektive und kein nachfolgendes Nomen* erkennt. Weitere Automaten existieren für die Personalpronomen, ein weiterer für Anreden wie *Herr Maier*. Für das Englische existieren entsprechende Automaten. Auf eine differenzierte Aufteilung innerhalb der NPs, wie es bei den meisten Grammatiken üblich ist, wird hier verzichtet, da dies in diesem Fall für die Übersetzung nicht notwendig ist.

Anschließend werden die PPs mit Hilfe der POS und unter Verwendung der bereits erkannten NPs extrahiert (siehe auch Abb. 4.3). Die PPs werden dabei grundsätzlich dem Verb zugeordnet. Für eine Entscheidung, ob die PP dem Verb oder einer NP zugeordnet werden soll, müßte entweder mehr Hintergrundwissen vorhanden sein (z.B. welche Präpositionen zu welchem Verb passen etc.), oder in den Beispielsätzen müßten die entsprechenden Zuordnungen angegeben sein, die zur Entscheidung herangezogen werden könnten. Allerdings muß diese Zuordnung manuell gemacht werden, was im Rahmen dieser Arbeit einen zu großen Aufwand erfordert hätte. Eine Analyse von typischen Beispielsätzen hat gezeigt, daß bei dieser Domäne eine PP-Zuordnung an eine NP recht selten ist.

Bei der Erkennung von Nebensatzgrenzen habe ich folgenden Ansatz verfolgt: Nur wenn mindestens 2 finite Verben vorhanden sind, werden Nebensatzgrenzen gesucht. Zuerst wird geprüft, ob Relativpronomen oder Konjunktionen in dem Satz vorkommen und entsprechend werden die Grenzen gezogen. Durch die Verbendstellung im Deutschen ist es recht einfach, das Ende des Relativ- bzw. Nebensatzes zu erkennen. Allerdings muß dazu der POS-Tagger finite und infinite Verben sicher unterscheiden können. Dies ist leider nicht immer der Fall. Die

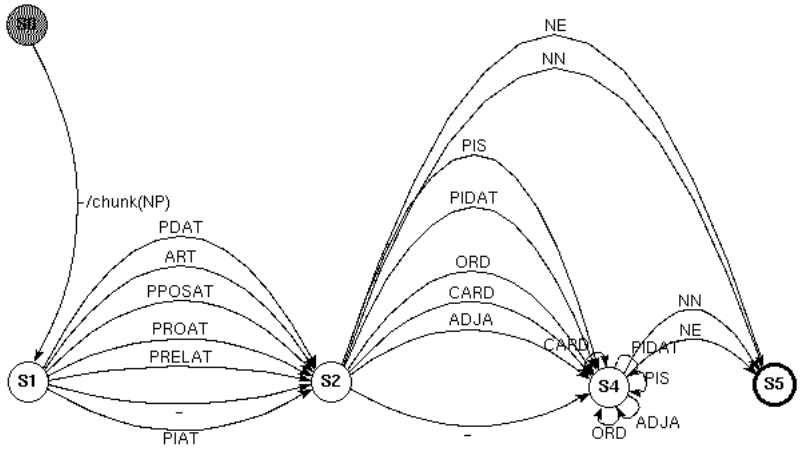

Abbildung 4.2: Ein NP-Erkennungsautomat

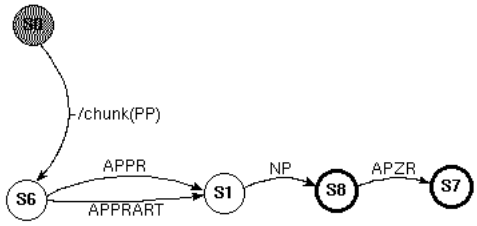

Abbildung 4.3: Ein PP-Erkennungsautomat

Relativsätze bzw. Nebensätze werden dabei, ähnlich wie NPs und PPs, in ein Objekt gepackt und mit S gelabelt (im folgenden S-Chunk genannt). Sind jedoch weder Relativpronomen noch Konjunktionen vorhanden, wird vor der ersten NP bzw. PP eine Satzgrenze gezogen, die vor dem zweiten finiten Verb steht. Untersuchungen am Beispielkorpus haben gezeigt, daß dies in den meisten – wenn auch nicht allen – Fällen ein korrektes Ergebnis liefert. Ein Beispiel dafür: *Ich glaube | ich habe am Dienstag keine Zeit* bzw. *Ich glaube | am Dienstag habe ich keine Zeit*. Um dieses Verfahren mit den Automaten effektiv realisieren zu können, ist eine Funktion `back()` implementiert, die auf dem Eingabestrom rückwärtsgehen kann. Der Automat ist in Abb. 4.4 dargestellt. Zur Erkennung von Satzgrenzen bei Sätzen wie *Soweit ich weiß | habe ich am Dienstag keine Zeit*, bei denen zwei finite Verben direkt hintereinander stehen, exisitiert noch ein zusätzlicher Automat (s. Abb. 4.5).

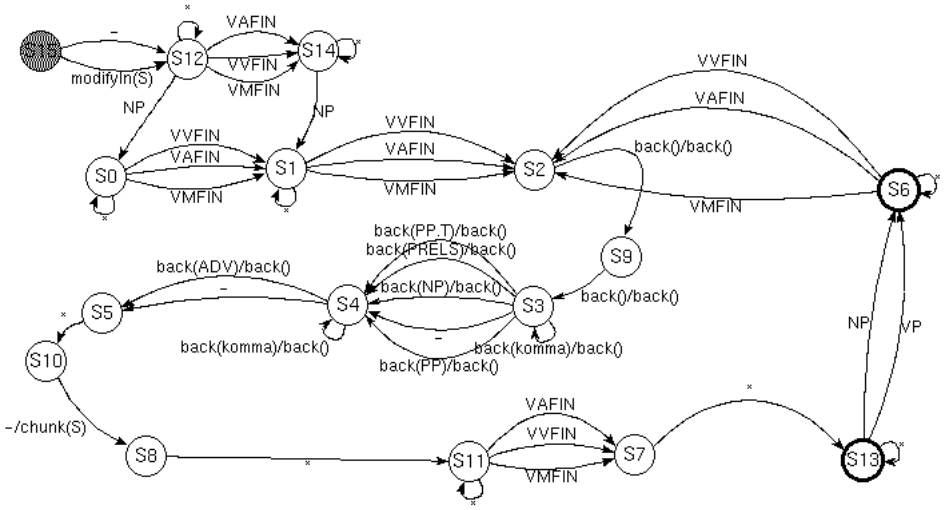

Abbildung 4.4: Ein Automat zur Erkennung von Satzgrenzen ohne Konjunktionen

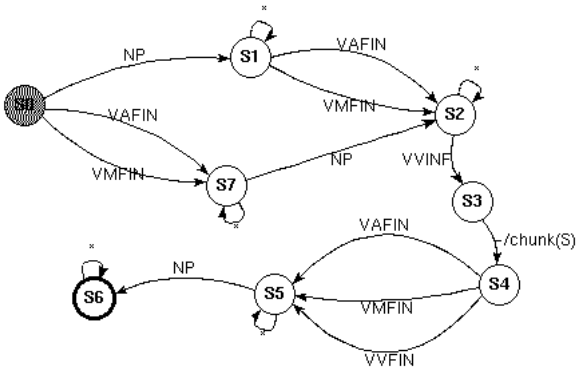

Abbildung 4.5: Automat zur Erkennung von Nebensatzgrenzen, wenn infinites und finites Verb direkt hintereinander stehen

Nachdem die Nebensatzgrenzen feststehen, werden die vom Verb abgetrennten Verbzusätze wieder dem Verb zugefügt, um später das Auffinden von Beipielsätzen zu ermöglichen (s.a. Abb. 4.6). Ein Beispiel: *Er kommt um 10 Uhr an* wird zu *Er ankommt um 10 Uhr*. Damit kann auch der Satz *Wenn er um 10 Uhr ankommt,...* korrekt übersetzt werden. Das Zusammenfügen der Wörter übernimmt dabei die Spezialfunktion `verb()`, wobei als Argument angegeben wird, ob es sich um das Verb (`verb`) oder den Zusatz (`pfix`) handelt. Um die notwendigen Änderungen auch in Nebensätzen durchführen zu können, ist die Funktion `modifyIn(`*Chunk-Name*`)` vorhanden. Diese Funktion ist notwendig, da Nebensätze bereits in Ob-

jekten abgespeichert sind und somit normalerweise nicht zugänglich sind. Diese Funktion ist relativ komplex aufgebaut: Falls das aktuelle Token auf dem Eingabestrom mit dem angegebenen Objekt-Namen übereinstimmt, wird der aktuelle Eingabestrom durch die Token, die in dem Objekt gespeichert sind, ersetzt. Der Ausgabestrom wird auf eine leere Liste umgeschaltet. Die ganze nachfolgend produzierte Ausgabe landet dann in dieser Liste. `Up` speichert am Ende die produzierte Ausgabe im Objekt ab (der alte Inhalt wird damit überschrieben) und schaltet wieder auf den normalen Eingabe- und Ausgabestrom zurück. Damit auch verschachtelte Nebensätze bearbeitet werden können, wird die Automatenebene mehrmals durchlaufen und bei jedem Durchlauf wird eine Ebene tiefer nach Objekten mit dem angegebenem Namen gesucht. So wird z.B. beim dritten Durchlauf in allen Objekten und deren Unterobjekten nach dem Objekt mit dem angegebenen Namen gesucht.

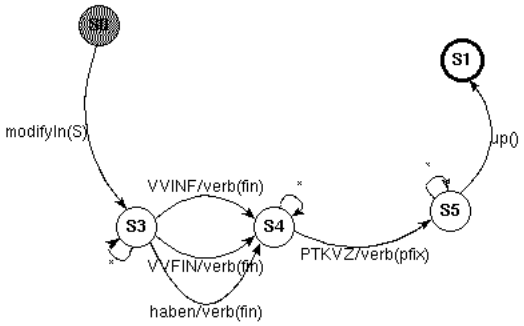

Abbildung 4.6: Der Automat zur Behandlung der abgetrennten Verbzusätze

4.5 Reparaturmaßnahmen

Einige Automaten bzw. Übergänge in den Automaten dienen dazu, vom POS-Tagger falsch bestimmte POS zu korrigieren und damit vor allem die Erkennung von Nebensatzgrenzen zu verbessern.

So erkennt der deutsche POS-Tagger irrtümlicherweise statt infiniten recht häufig finite Verben, obwohl noch ein Hilfsverb im Satz vorhanden ist. Dies wird mit einem speziellen Automaten korrigiert (siehe Abb. 4.7). Somit wird verhindert, daß andere Automaten eine Satztrennung zwischen beiden Verben vornehmen, die sonst stattfinden würde. Manchmal führt dies zwar zu Fehlern, da es sich bei dem Hilfsverb eigentlich um ein Vollverb handelt und somit sehr wohl eine Satztrennung stattfinden sollte, aber der positive Effekt überwiegt bei weitem, wie Tests gezeigt haben.

Im Englischen werden einige Verben häufig als Nomen erkannt (z.B. *suit*, *need*). In den Beispielsätzen tauchen diese Wörter jedoch nur in Form von Verben auf

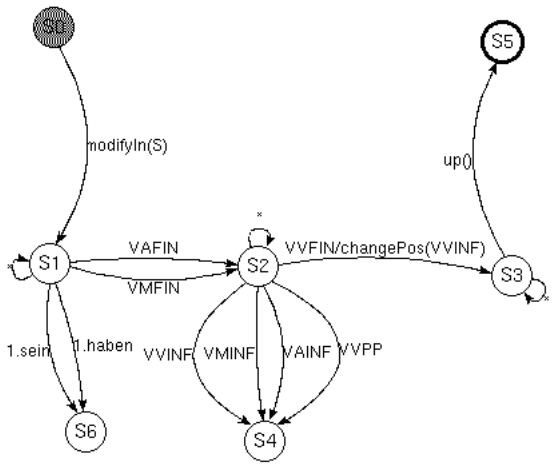

Abbildung 4.7: Aus finiten Verben werden infinite, wenn ein Hilfsverb im Satz vorkommt.

und daher wird die POS dieser Wörter geändert. Einige Konjunktionen werden ebenfalls nicht immer als solche erkannt und geändert, beispielsweise *aber*, *because* und *if*. Sie werden zwar nicht immer im Sinne von Konjunktionen eingesetzt, es ist aber günstig, sie immer als Konjunktionen zu betrachten, da dort potentiell Satzgrenzen sein können und die Automaten zur Nebensatzerkennung stets abprüfen, ob in jedem Nebensatz ein Verb vorhanden ist. Um die Gewichtung später in der Transferkomponente zu verbessern, werden noch infinite Verben, die allein in einem Satzteil vorkommen, in finite Verben umgewandelt.

Im Deutschen werden Präpositionen mit integriertem Artikel (wie z.B. *am*, *ins*) in Präposition und Artikel zerlegt, damit eine klare Auftrennung in PP und NP möglich ist und die Wörter einzeln übersetzt werden können. Dazu besteht die Aktion `word`, die neue Wörter zusammen mit der POS und der Stammform auf den Ausgabestrom schreiben kann.

Um Fehler bei dem POS-Tagging auszugleichen, wird zusätzlich bei der Erkennung von Nebensatzgrenzen vorausgesetzt, daß mindestens eine NP im Satz vorkommt. Eine weitere Einschränkung ist, daß keine Nebensatzgrenze vor einer Konjunktion gezogen wird, wenn auf diese Konjunktion direkt ein Verb folgt (siehe auch Automat in Abb. 4.5).

Für einige feststehende Ausdrücke, die kurze Nebensätze darstellen (wie z.B. im Deutschen *glaube ich*, *würde ich sagen* bzw. im englischen *I have to say*) sind noch separate Automaten vorhanden, die die Satzgrenzen entsprechend erkennen. Sie werden aufgrund ihrer Kürze häufig von den restlichen Automaten nicht korrekt erkannt (z.B. werden davorstehende NPs noch versehentlich dazugenommen).

4.6 Ausblick

Eine vollständigere Abdeckung der Grammatik wäre wünschenswert, wobei folgende Phänomene am nötigsten gebraucht werden: die Erkennung von Infinitivsätzen, die Entscheidung, ob eine PP einer NP oder dem Verb zugeordnet werden soll, Behandlung von *und* bzw. *oder*-Konjunktionen auf NP-Ebene durch zusätzliche Knoten. Damit die Grenzen der Infinitivsätze im Deutschen überhaupt erkannt werden können und für die Entscheidung, ob ein PP-Attachment an eine NP stattfinden soll, ist es notwendig, auf die semantische Datenbank von VERBMOBIL zurückzugreifen, in der u.a. abgespeichert ist, von welchem semantischen Typ die Argumente des Verbs sein dürfen.

Für eine anspruchsvollere syntaktische Analyse ist allerdings eine nicht-deterministische Grammatik unabdingbar, da viele Ambiguitäten nicht auf rein syntaktischer Ebene aufgelöst werden können. Dies kann durch die Verwendung nicht-deterministischer Automaten erreicht werden oder indem die einzelnen Automaten zwar deterministisch arbeiten, aber ein- und ausgeschaltet werden können. Um aus den verschiedenen Syntaxbäumen den richtigen auszuwählen, könnte überprüft werden, wie gut sie zu den Beispielsätzen passen.[1]

Durch gezieltes Ein- bzw. Ausschalten von Automaten könnte in diesem Fall die Anzahl der generierten Syntaxbäume reduziert werden. Ein häufig auftretendes Problem bei Grammatiken besteht darin, daß alle möglichen Syntaxbäume generiert werden. Wenn z.B. zweimal im Satz unklar ist, ob eine PP dem Verb oder einer NP zugeordnet wird, werden vier verschiedene Syntaxbäume generiert. Bei den Automaten könnte bei der ersten unklaren Stelle die entsprechenden Automaten einmal ein- und ausgeschaltet werden. Anhand der Beispielsätze wird dann ermittelt, welche Variante wahrscheinlicher ist. Für die zweite unklare Stelle wird dieser Vorgang wiederholt. Es müssen somit nur zwei, anstelle von vier Syntaxbäumen untersucht werden.

Die hierarchischen Automaten können natürlich auch anderweitig eingesetzt werden, so kommen sie in der VERBMOBIL-Dialoggruppe bereits für folgende Aufgaben zum Einsatz:
Natürlichsprachliche Zeitangaben werden in formale, logische Ausdrücke der Zeitbeschreibungssprache TEL[End98] umgewandelt (bzw. wird auch Sprache aus TEL-Ausdrücken erzeugt). Inzwischen werden auch komplexere semantische Entitäten aus sprachlichen Äußerungen extrahiert. Im Rahmen des flachen Verarbeitungsstrangs von VERBMOBIL werden sogar ganze Äußerungen von Automaten direkt übersetzt.

[1] Die Beispielsätze müssen dann allerdings eventuell manuell überarbeitet werden, damit dort die richtige syntaktische Struktur vorhanden ist. Dies ist notwendig, da natürlich auch bei den Beispielsätzen mehrere alternative Syntaxbäume auftreten können.

Kapitel 5

Template-Generierung

5.1 Aufgabe der Template-Generierung

Die Hauptaufgabe der Template-Generierung besteht darin, für die Wörter in den Beispielsatzpaaren die entsprechenden Übersetzungen zu finden und zuzuordnen (auch Alignment genannt). Dabei können nicht nur Wortpaare gebildet, sondern auch gleich mehrere Wörter zusammengefaßt werden, die das Transfermodul dann zusammen übersetzt, z.B. *hat keine Zeit* → *is not free*. Eine weitere Aufgabe liegt darin, Knotenbewegungen festzustellen, die das Transfermodul beim Übersetzen ausführen muß, z.B. Subjekt-Objekt-Vertauschungen oder Objekt-PP-Wechsel, und diese Knotenbewegungen an ein Wort zu binden, beispielsweise an ein Verb. So muß beim Übersetzen des Verbs *treffen* in dem Satz *Ich treffe mich mit jemanden* aus der Präpositionalphrase *mit jemanden* im Englischen ein direktes Objekt werden: *I meet somebody*.

Beim Alignment stellt sich grundsätzlich die Frage, welche Wörter in den Beispielsatzpaaren zusammengelinkt werden sollen. So können in dem Beispielsatzpaar

ich habe keine Zeit → *I have no time*

die Wortpaare *ich* → *I*, *haben* → *have*, *keine* → *no* und *Zeit* → *time* gebildet werden. Es ist aber auch möglich, das ganze als feststehenden Ausdruck zu betrachten und nur die Paare *ich* → *I* und *keine Zeit habe* → *have no time* zu bilden. Im ersten Fall kann unter Verwendung des Beispielsatzpaares auch in einem anderen Kontext *Zeit* mit *time* übersetzt werden, im zweiten Fall ist das nicht möglich. Trotzdem kann es durchaus sinnvoll sein, größere Einheiten zusammenzufassen. So ist es bei dem Satz

es ist schlecht am Donnerstag → *it is not convenient on Thursday*

nicht sinnvoll, *schlecht* mit *not convenient* zu linken, da die Übersetzung normalerweise *bad* ist. Wenn im genannten Beispiel trotzdem *schlecht* mit *not convenient* zusammengefaßt wird, hängt es von der Auswahlfunktion ab, ob aus den

Beispielsatzpaaren, die eine Übersetzung von *schlecht* enthalten, die passende herausgefunden wird. Da beim Alignment Fehler auftreten können, steigt außerdem das Risiko, bei kleineren Einheiten unpassende Wörter zusammenzulinken.

Zusammenfassend gilt:

- Je mehr Wörter in einen Link gepackt werden, desto größer ist das Risiko, daß ein zur Übersetzung eines Wortes eigentlich passender Beispielsatz übergangen wird, weil ein Wort im zu übersetzenden Satz fehlt oder anders ist als im Beispielsatz.
- Bei wenigen Wörtern pro Link steigt das Risiko, daß zur Übersetzung eines Wortes ein unpassender Beispielsatz genommen wird und die Übersetzung damit falsch wird, weil die Bewertungsfunktion, die den Beispielsatz auswählt, nicht gut genug arbeitet.

Bei meinem System wird versucht, die Anzahl der Wörter pro Link möglichst gering zu halten, um flexibel auf die zu übersetzenden Sätze reagieren zu können, darauf hoffend, daß nicht zu viele Fehler auftreten und die Auswahlfunktion hinreichend gut ist.

Nachdem die Auswirkungen, die von der Größe der zusammengefaßten Einheiten abhängen, erläutert wurden, folgt nun eine Beschreibung, wie die Zuordnung im einzelnen funktioniert. Die prinzipielle Vorgehensweise bei dem hier verwendeten Alignment ist dem HTG-Vorschlag [KKM92] ähnlich. Im Unterschied zu HTG wird jedoch zum Koppeln der Wörter nicht nur ein bilinguales Wörterbuch verwendet, sondern auch die POS der Wörter und die Satzstruktur werden stärker berücksichtigt. Aus diesem Grund können im Wörterbuch nicht vorkommende Wörter trotzdem zusammengelinkt werden. Dies hat den Vorteil, daß bei neuen Domänen mit speziellem Wortschatz das Wörterbuch in einem wesentlich geringeren Ausmaß ergänzt werden muß.

Die Überlegung bei der Kopplung von Wörtern mit Hilfe ihrer POS ist, daß ausgangssprachliche und zielsprachliche Wörter mit vergleichbarer POS wörtliche Übersetzungen sind. Welche der POS jeweils vergleichbar sind, ist in einer Tabelle gespeichert. So entsprechen der deutschen POS für Adverbien *ADV* die englischen POS *RB,JJ*. Zu beachten ist, daß sowohl im ausgangs- als auch im zielsprachlichen Satz nicht mehr als ein ungelinkes Wort mit der gleichen POS vorkommen darf, da sonst die Wörter und deren Übersetzungen nicht mehr eindeutig zugeordnet werden können. Tests haben gezeigt, daß die Berücksichtigung der Reihenfolge der Wörter als Auswahlkriterium zwischen mehreren Alternativen beim Linken zu einer relativ hohen Fehlerquote führt.

Eine weitere Ergänzung gegenüber dem HTG-Ansatz besteht darin, diejenigen NPs und PPs, die nicht mit Hilfe des Wörterbuchs gekoppelt werden konnten, ebenfalls unter Verwendung der POS zu linken. Voraussetzung dafür ist, daß die POS in beiden Phrasen vergleichbar sind, d.h. für jedes Wort in der Zielsprache muß ein entsprechendes Wort in der Ausgangssprache vorhanden sein. Im Gegensatz zur Wortkopplung müssen die Phrasen nicht eindeutig zuordbar sein.

Wie Tests gezeigt haben, sind die Ergebnisse sehr gut, wenn die Phrasen der Reihenfolge nach gekoppelt werden.

Zusätzlich werden auch noch NPs bzw. PPs gekoppelt, wenn in jeder Sprache nur noch eine bislang unkoppelte Phrase vorhanden ist.

Das Wörterbuch kann im Gegensatz zum HTG-System nicht nur einzelne Wortpaare enthalten, sondern auch mehrere Wörter zusammen mit syntaktischer Struktur (z.B. *lassen [NP Sie] [NP uns]* $\leftrightarrow$ *let [NP us]*). Außerdem besteht somit die Möglichkeit, das Wörterbuch iterativ mit bislang extrahierten Kopplungen zu erweitern. Auf eine Implementierung wurde im Rahmen dieser Arbeit aus Zeitgründen verzichtet.

Im Unterschied zum HTG-Vorschlag wird das Koppeln von Knoten nicht zur syntaktischen Desambiguierung genutzt, da die syntaktische Analyse ohnehin nur einen Syntaxbaum erzeugt. Ein weiterer Unterschied besteht darin, daß keine Translation Templates erzeugt werden, sondern die Speicherung der Kopplungen direkt in den Beispielsatzpaaren erfolgt. Dies erlaubt eine kompaktere Speicherung, stellt aber keinen funktionalen Unterschied dar.

5.2 Alignment-Algorithmus

Zunächst findet, wie auch beim HTG-Ansatz, eine syntaktische Vorverarbeitung der Beispielsatzpaare statt. Die POS wird mit Hilfe der beschriebenen POS-Tagger sowohl für die ausgangs- als auch zielsprachlichen Wörter in den Beispielsatzpaaren bestimmt. Anschließend werden die Sätze in der Ausgangs- und Zielsprache mit den in Kapitel 4 beschriebenen hierarchischen Automaten syntaktisch analysiert.

Die Ergebnisse des Alignments können mit einem Tcl/Tk-Tool graphisch veranschaulicht werden. Die nachfolgenden Abbildungen der Beispiele sind direkte Bildschirmabzüge dieses Tools.

Die beiden Hauptfunktionen der Alignmentkomponente sind nachfolgend noch als Pseudo-Code dargestellt. Die Funktion `alignment` nimmt dabei als Argumente die beiden analysierten Sätze und ruft zuerst `alignWordsPerDictionary` (2) auf, um die Wörter mittels Wörterbuch zu koppeln. Die nicht gekoppelten Wörter und die Knoten werden mit dem Aufruf von `alignChunk` (3) versucht, sinnvoll zu koppeln. Die Funktionen `alignWordsPerDictionary`, `alignChunk` und die darin aufgerufenen Funktionen werden anschließend einzeln vorgestellt und ausführlich beschrieben. Zusätzlich sind in Abb. 5.1 und Abb. 5.2 zur Illustration der einzelnen Routinen noch zwei Beispiele dargestellt.

Noch eine Bemerkung zur der Variablenbenennung: `g` und `e` bezieht sich auf den ganzen ausgangs- bzw. zielsprachlichen Satz (`g` für *german* und `e` für *english*), bei `gc` und `ec` kann es sich auch nur um Satzteile handeln (`c` steht für *chunk*).

```
( 1)  proc alignment(g,e)
( 2)    alignWordsPerDictionary(g,e)
( 3)    alignChunk(g,e)
( 4)  end

( 5)  proc alignChunk(gc,e)
( 6)    alignedChunkList = [ ]
( 7)    forall chunks sub_gc in gc do
( 8)      alignedChunkList.append(alignChunk(sub_gc,e))
( 9)    end
(10)    ec = searchChunk(g,e)
(11)    if ec != NULL then
(12)      alignSubChunkPerPOS(gc,ec)
(13)      alignWordsPerPOS(gc,ec)
(14)      alignChunkSameLevel(ec,gc)
(15)      alignRest(gc,ec)
(16)      return alignChunksWithWords(gc,ec,alignedChunkList)
(17)    else
(18)      return alignedChunkList
(19)  end
```

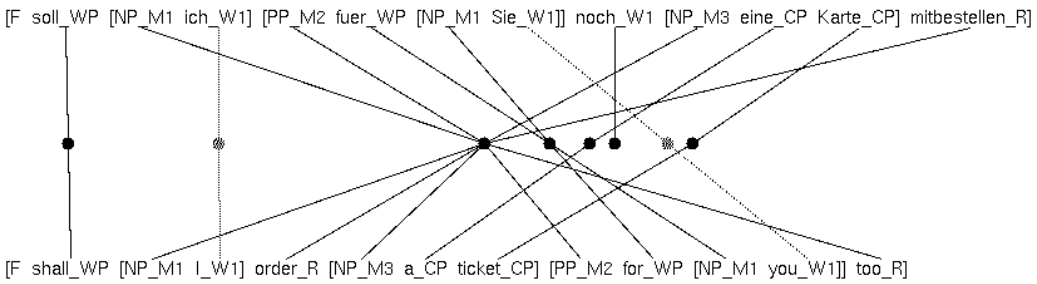

Abbildung 5.1: Kopplung aus dem verwendeten Koprus (1)

alignChunk(gc,e)

Diese Funktion hat die Aufgabe, die noch nicht übers Wörterbuch gekoppelten Wörter zu linken und für die einzelnen ausgangssprachlichen Knoten zielsprachliche Entsprechungen zu finden. Dazu läuft die Funktion bottom-up durch den deutschen Syntaxbaum (7)-(9). Falls für den deutschen Knoten `gc` eine englische Entsprechung mit Hilfe der Funktion `searchChunk` (10) gefunden wurde, wird nun versucht, die innerhalb der beiden Knoten liegenden noch nicht gelinkten Wörter zu koppeln. Dabei kommen folgende Funktionen zum Einsatz (12)-(15):

`alignSubChunkPerPOS`
: Unterknoten werden mittels POS gelinkt

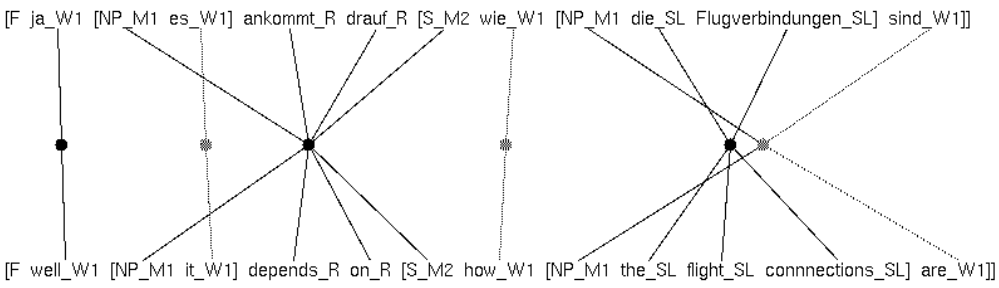

Abbildung 5.2: Kopplung aus dem verwendeten Korpus (2)

`alignChunkSameLevel`
: Unterknoten werden aufgrund ihrer singulären Vorkommensweise gelinkt

`alignWordsPerPOS`
: Wörter werden mittels POS gelinkt

`alignRest`
: Verarbeitet die übriggebliebenen ungelinkten Wörter und Knoten

Mit der Funktion `alignChunksWithWords` (16) wird versucht, die bereits in rekursiven Funktionsaufrufen gelinkten Knoten an einzelne Wörter zu koppeln (z.B. ein NP-Knoten an ein Verb). Knoten, die nicht mit einem Wort gekoppelt werden konnten, werden als Resultat der Funktion `alignChunk` zurückgeliefert.

alignWordsPerDictionary(gc,ec)

Die einzelnen Wörter des deutschen Satzes werden im bilingualen Wörterbuch gesucht. Dazu wird die gleiche Funktion (`findBestTranslation`, S. 61) verwendet, die auch bei der Verwaltung der Beispielsätze angewandt wird. Der Vorteil liegt darin, daß mehrere Wörter mit syntaktischer Struktur in das Wörterbuch aufgenommen werden können. Das Wörterbuch selbst wurde manuell von mir erstellt und enthält 300 Einträge, darunter 30 Eintäge, die aus mehreren Wörtern bestehen. Das Wörterbuch enthält dabei kaum speziellen Wortschatz, sondern fast nur domänenunabhängige Wörter. Domänenspezifische Wörter und sämtliche Verben werden durch den Alignment-Algorithmus gekoppelt.

Wenn eine der abgespeicherten Übersetzungen im zielsprachlichen Satz gefunden wird, erfolgt die Kopplung. Auf die Beschränkung, daß die Wörter nur dann gekoppelt werden, wenn sie eindeutig gekoppelt werden können, wie dies beim HTG-Vorschlag der Fall ist, wurde hier verzichtet. Stattdessen ist die Reihenfolge der Wörter entscheidend, um bessere Alignment-Resultate zu erzielen (s.a. Kap. 8). In den Beispielen sind die über das Wörterbuch gekoppelten Wörter mit einem **W** versehen.

Eine Besonderheit des erstellten Wörterbuchs ist die Möglichkeit, für bestimmte ausgangssprachliche Wörter keine zielsprachliche Übersetzung anzugeben. Dies ist bei vielen Füllwörtern notwendig, wie z.B. *dann, also, nun*, die häufig im zielsprachlichen Beispielsatz einfach wegfallen und nicht übersetzt sind. Diese Wörter bekommen, falls sonst keine der angebenen Übersetzungen im zielsprachlichen Teil vorkommen, einen einseitigen Link ohne zielsprachliches Wort. Wenn zum Übersetzen des Wortes dann dieses Beispielsatzpaar ausgewählt wird, wird kein Wort generiert.

Im Wörterbuch selbst sind außer Hilfsverben keine Verben verzeichnet, da diese über die POS richtig gelinkt werden, vorausgesetzt die Nebensätze sind richtig zugeordnet. Tests haben gezeigt, daß dies bei den verwendeten Beispielsatzpaaren fast immer zutrifft.

searchChunk(gc,e)

Für einen ausgangssprachlichen Knoten `gc` wird versucht, den entsprechenden Knoten im zielsprachlichen Satz `e` zu finden, also z.B. für die ausgangssprachliche NP, die das Subjekt des Satzes bildet, die entsprechende zielsprachliche NP. Voraussetzung dafür ist, daß Wörter innerhalb der Knoten nicht durch einen Link mit Wörtern außerhalb der Knoten verbunden sind (äquivalentes Vorgehen zu HTG-Vorschlag, siehe auch Abb. 5.3). Weitere Bedingungen sind, daß mindestens ein Wort der Ausgangssprache mit einem Link gekoppelt sein muß und daß die Knotenbezeichnungen übereinstimmen müssen, d.h. daß z.B. NP-Knoten nicht mit S-Knoten gelinkt werden können. Test haben gezeigt, daß ein solcher Link meist durch Fehler beim POS-Taggen bzw. der syntaktischen Analyse zustandekommt und häufig die Übersetzungsqualität beeinträchtigt.

Der am tiefsten liegende Knoten im zielsprachlichen Syntaxbaum, der die Bedingungen erfüllt, wird als Resultat zurückgegeben. Falls kein entsprechender Knoten gefunden wird, wird als Ergebnis *NULL* zurückgeliefert.

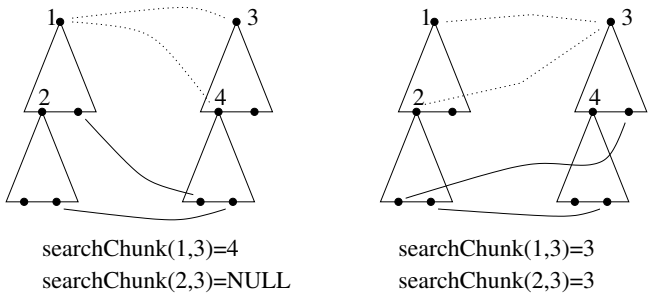

Abbildung 5.3: Zwei schematische Beispiele für Vorgehensweise bei `searchChunk`. Die durchgezogenen Linen stellen Wortkopplungen dar, die gestrichelten Linien mögliche Kopplungen zwischen Knoten

alignWordsPerPOS(gc,ec)

Diese Funktion hat die Aufgabe, Wörter, die direkt in den Knoten `gc` und `ec`, d.h. nicht in Unterknoten, enthalten sind und bisher noch nicht gelinkt werden konnten, mit Hilfe der POS zu koppeln. Dazu wurde manuell eine Tabelle erstellt (siehe 5.1), die für die ausgangssprachlichen POS die zielsprachlichen Entsprechungen enthält. Das hat den Vorteil, daß durch die Kopplung mit den POS diejenigen Wörter gekoppelt werden können, die nicht im Wörterbuch verzeichnet sind. Durch diesen Schritt können später Wörter auch gesondert in einem veränderten Kontext übersetzt werden. Um Fehler zu vermeiden, müssen die Wörter eindeutig zuordbar sein, d.h. nur wenn jeweils ein Wort dieselbe POS hat, wird gelinkt. Ein Beispiel dafür: Soll ein ungekoppeltes Adverb mit der POS *ADV* gekoppelt werden, darf in `gc` kein weiteres ungekoppeltes Wort mit den POS aus der gleichen Gruppe (also hier *ADJD*, *ADJA*, *ITJ*, *ADV*) vorkommen und exakt eines in `ec` aus der Gruppe (also *JJ*, *RB*, *EX*). Die Einteilung in Gruppen ist notwendig, da meist keine eins-zu-eins-Umsetzung der ausgangssprachlichen POS in zielsprachliche POS existiert. Außerdem arbeitet der POS-Tagger nicht immer korrekt, so daß es günstiger ist, die Gruppen nicht zu strikt abzugrenzen. In den Beispielen sind die Links mit **WP** gekennzeichnet.

deutsche POS	engl. POS	deutsche POS	engl. POS
ART PIAT	DT	NN	NN NNS
ADJD ADJA ITJ ADV	JJ RB EX	APPR APZR APPRART	IN
PIDAT	DT	PPOSAT	PRP$
PPER	WDT	PDS	WDT
PRF	PRP	VAFIN VMFIN	MD
VAFIN VVFIN	VBZ		

Tabelle 5.1: deutsche POS und die englischen Entsprechungen, die Zuordnung wird in den Funktionen `alignWordsPerPos` und `alignSubChunkPerPOS` eingesetzt

alignSubChunkPerPOS(gc,ec)

Mittels dieser Funktion sollen anhand der POS die unmittelbaren Unterknoten von `gc` gelinkt werden, bei denen die Kopplung mit Hilfe der Funktion `searchChunk` nicht möglich war. Während `alignWordsPerPOS` nur einzelne Wörter koppelt, werden hier Knoten zusammen mit den darin enthaltenen Wörtern und Unterknoten gekoppelt. Ein Beispiel:

[PP an (APPR) [NP dem (ART) Flughafen (NN)]] →
[PP at (IN) [NP the (DT) airport (NN)]]

Die POS ist in diesem Fall in Klammern angegeben und keines der Wörter ist im Wörterbuch enthalten.

Mit Hilfe der Funktion `alignWordsPerPOS` wird überprüft, ob für jedes ausgangssprachliche Wort im Unterknoten exakt ein zielsprachliches Wort mit korrespondierender POS existiert. Test haben gezeigt, daß diese relativ strikte Bedingung nötig ist, da sonst zu viele inkorrekte Kopplungen auftreten[1]. Der Aufruf von `alignWordsPerPos` koppelt im Erfolgsfall auch gleich die Wörter einzeln miteinander, also in dem genannten Beispiel *an* mit *at*, *dem* mit *the* und *airport* mit *Flughafen*. In den Abb. 5.1 und 5.2 sind diese Links mit **CP** gekennzeichnet.

Mögliche Gründe warum die Funktion `searchChunk` eigentlich zusammenpassende Knoten nicht koppeln kann:

- kein Wort aus dem Knoten wurde im Wörterbuch gefunden
- kein Wort konnte eindeutig zugeordnet werden
- ausgangssprachliche Wörter wurden "aus Versehen" mit zielsprachlichen Wörtern gekoppelt, die außerhalb des Knotens stehen, da das korrespondierende Wort im gleichen Knoten nicht im Wörterbuch vorkommt

alignChunkSameLevel(ec,gc)

Diese Funktion koppelt ähnlich wie `alignSubChunkPerPos` unmittelbare Unterknoten von `gc` und `ec`, die bisher noch nicht gelinkt werden konnten. Allerdings müssen die Wörter in beiden Phrasen nicht wie in `alignSubChunkPerPos` entsprechende POS haben.

Damit die Unterknoten gekoppelt werden können, müssen beide Knoten die gleiche Bezeichnung (z.B. *NP*) tragen und alle restlichen Unterknoten mit derselben Bezeichnung müssen bereits gekoppelt sein. Um brauchbare Ergebnisse zu erhalten, dürfen die Wörter in den Unterknoten nur bestimmte POS haben, da sonst bei freieren Übersetzungen – wie Experimente gezeigt haben – zu viele Kopplungen zwischen Phrasen auftreten, die nichts miteinander zu tun haben. Eine Liste mit den zulässigen POS findet sich in Tab. 5.2. Im Gegensatz zu `alignSubChunksPerPos` können die Wörter nicht einzeln gelinkt werden, deshalb werden sie alle zusammengekoppelt. Die Kennzeichnung in den Bespielen ist **SL**.

deutsche POS	englische POS
PDAT PPOSAT PROAT PRELAT	DT PRP$ RB JJ JJS JJR
ART ADJD PIAT CARD ORD	CD NN NNS NNP
PIS PIDAT NN NE	

Tabelle 5.2: zulässige deutsche und englische POS in `alignChunkSameLevel`

[1] Eine getestete Variante war, die Übereinstimmung der POS auf lediglich 60% zu beschränken

alignRest(gc,ec)

Diese Funktion hat die Aufgabe, Wörter und Knoten innerhalb von `gc` und `ec` zu koppeln, die bisher noch nicht gelinkt werden konnten. Dazu gehören auch solche Knoten, für die bisher keine ausgangs- bzw. zielsprachliche Entsprechung gefunden wurde, inklusive aller darin enthaltener nicht gelinkter Wörter. Da das Wörterbuch keine Verben enthält und Verben auch nicht mittels POS gekoppelt werden, erfolgt das Linken von Verben erst in dieser Funktion. Die Überlegung dabei ist, daß diejenigen Wörter, die einander zugeordnet werden können, vorher bereits zusammengelinkt wurden. Der Rest wird dann mit dem Verb gekoppelt. Dafür ein Beispiel:

das anhört sich gut → *that sounds good*

bei dem *das* mit *that* und *gut* mit *good* mittels Wörterbuch gekoppelt wurde und nun in `alignRest` die restlichen Wörter *anhört sich* und *sounds* gelinkt werden. Somit wird *sich* und *anhört* später zusammen als *sounds* übersetzt. In den Abb. 5.1 und 5.2 sind die Wörter, die mittels `alignRest` gekoppelt wurden, mit einem **R** gekennzeichnet.

Probleme treten auf, wenn vor dem Aufruf der Funktion alle ausgangssprachlichen Wörter gekoppelt sind, aber nicht alle zielsprachlichen. In diesem Fall fehlen den zielsprachlichen Wörtern ausgangssprachliche Wörter, mit denen sie gelinkt werden können. Sie müssen daher den bereits gelinkten Wörtern zugefügt werden. Der umgekehrte Fall, bei dem nur noch ungelinkte ausgangssprachliche Wörter vorhanden sind, ist weniger problematisch, sie werden einfach nicht übersetzt.

alignChunkWithWords(gc,ec,alignedChunkList)

Diese Funktion hat nicht das Ziel, Wörter oder Knoten zu koppeln, sondern es geht darum, daß die in `searchChunk` gefundenen Entsprechungen von ausgangs- und zielsprachlichen Knoten festgehalten sind, indem sie an ein Wort gekoppelt werden. Bei NP- und PP-Knoten auf Satzebene wird versucht, sie an ein finites Verb zu koppeln. Bei in PPs geschachtelten NPs werden die Knoten an eine Präposition gekoppelt. Wenn kein entsprechendes Wort vorhanden ist, wird eine Liste mit den Knoten als Ergebnis zurückgegeben, um sie eine Ebene höher zu koppeln. Wenn keine höhere Ebene vorhanden ist, werden sie an irgendein anderes Wort (falls vorhanden) gekoppelt.

Die Kopplung der Knoten an ein Wort ist notwendig, weil sonst die Knotenverschiebungen (z.B. Vertauschung von Subjekt und Objekt) nicht funktionieren. Die Ursache liegt darin, daß im Transfermodul Knoten des Beispielsatzes entsprechende Knoten im zu übersetzenden Satz zugeordnet werden. Beim Übersetzen der Knoten des zu übersetzenden Satzes werden diese an die entsprechende Stelle verschoben. Eine ausführliche Beschreibung dazu findet sich in Abschnitt 6.4.

addToIndex(g,e)

Um zu vermeiden, daß später beim Übersetzen nicht alle Beispielsätze in den Hauptspeicher geladen werden müssen, wird eine Indextabelle aufgebaut. Als Schlüssel enthält sie eine Liste aller Wörter, die zusammen übersetzt werden müssen. Dazu wird die Position in der Datei abgespeichert, die die Beispielsatzpaare enthält. Für jeden Link ist damit ein eigener Eintrag vorhanden, d.h. jeder Beispielsatz hat normalerweise mehrere Einträge. Die Verwendung einer Indexdatei verkürzt wesentlich die Startzeit des Programms. Dies ist bei der Entwicklung mit einer Sprache, in der die Programme compiliert werden müssen, sehr zeitsparend, da nach jeder Änderung das Programm neu gestartet werden muß.

5.3 Ausblick

Die beschriebene Vorgehensweise funktioniert nur bei einer kleinen Anzahl oder bei speziell ausgewählten Beispielsätzen. Um einen umfangreichen satzweise alignierten bilinugalen Korpus mit viel Redundanz verarbeiten zu können, müssen Beispielsatzpaare mit gleichen Übersetzungen zusammengefaßt werden. Ein Problem dabei ist, wie der Kontext einbezogen werden kann. Normalerweise sind die Wörter aus dem Kontext ja verschieden. Entweder werden einige Beispielsätze mit ähnlichem Umfeld aussortiert (dazu muß jedoch eine Ähnlichkeitsfunktion für das Umfeld existieren) oder die Wörter werden mit Attributen versehen. Diese Attribute werden statt der ursprünglichen Wörter verwendet und dadurch wird eine Reduzierung der Beispielsatzpaare erreicht. Attribute haben den weiteren Vorteil, daß später bei der Auswahl des passenden Beispielpaares *decision trees* verwendet werden können [HM97].

Bei nicht manuell erstellten Beispielkorpera sollte die Übersetzungsqualität der einzelnen Satzpaare bewertet werden können. Damit ließen sich z.B. sehr freie Übersetzungen entfernen bzw. würden dort Abweichungen im Kontext zu stärkeren Abwertungen als im Normalfall führen.

Auch denkbar wäre der Einsatz einer solchen Funktion, um die manuelle Überprüfung des Alignments zu unterstützen, so daß nur ein Teil der Satzpaare manuell geprüft werden muß.

Eine Erweiterungsmöglichkeit ist der schrittweise Ausbau des Wörterbuchs mit gekoppelten Wörtern, die bislang nicht im Wörterbuch stehen, die jedoch durch die Alignment-Funktion gekoppelt werden konnten. So ist denkbar, das Alignment immer wieder zu starten, jedesmal mit einem umfangreicheren Wörterbuch. Allerdings ist natürlich die Gefahr vorhanden, falsche Einträge aufzunehmen und damit Folgefehler zu produzieren.

Kapitel 6

Transfer und Generierung

6.1 Überblick

Die Transferkomponente hat die Aufgabe, den bereits von der syntaktischen Analyse erzeugten ausgangssprachlichen Syntaxbaum in einen zielsprachlichen Syntaxbaum mit Hilfe der alignierten Beispielsatzpaare umzuwandeln. Die Generierungskomponente sortiert die Wörter in dem ausgangssprachlichen Syntaxbaum so, daß die Wörter im Baum direkt von links nach rechts ausgegeben werden können.

Prinzipiell arbeitet die Transferkomponente so, daß die Wörter aus dem ausgangssprachlichen Syntaxbaum in eine Liste geschrieben werden. Diese Liste mit zu übersetzenden Wörtern wird wortweise abgearbeitet, indem für jedes Wort nach der geeignetsten Übersetzung in den Beispielsatzpaaren mit Hilfe einer Auswahlfunktion gesucht wird. Die Tatsache, daß auch mehrere Wörter zusammen übersetzt werden können, wie z.B. in

ich habe keine Zeit → *I am not free*

wobei *habe keine Zeit* zusammen zu *am not free* übersetzt wird, hat Konsequenzen. So werden nur Beispielsatzpaare bei der Auswahl berücksichtigt, in der alle Wörter, die gleichzeitig mit dem aktuell zu übersetzenden Wort übersetzt werden, in der Liste mit den zu übersetzenden Wörtern vorkommen. Danach werden aus der Liste mit den zu übersetzenden Wörtern alle Wörter gestrichen, die zusammen übersetzt wurden, damit diese nicht mehrmals übersetzt werden. Eine Konsequenz aus dieser Vorgehensweise ist, daß die Reihenfolge der Wörter in der Liste mit den zu übersetzenden Wörtern u.U. Einfluß auf das Übersetzungsresultat haben kann. So kann z.B. das obige Beispielsatzpaar nicht mehr angewendet werden, wenn *Zeit* bereits früher als Einzelwort übersetzt wurde.

Daher werden die Wörter aus dem ausgangssprachlichen Satz nicht einfach von links nach rechts in die Liste geschrieben, sondern die Liste wird vorher noch umsortiert. Eine Möglichkeit wäre gewesen, die POS der einzelnen Wörter zu berücksichtigen und z.B. in der Liste Verben vor Adjektiven einzuordnen. Es

wurde allerdings ein anderer Ansatz gewählt: Die Liste wird aufsteigend nach der Anzahl der Beispielsatzpaare, die für die einzelnen Wörter vorhanden sind, sortiert. Das basiert auf der Annahme, daß im Satz inhaltlich wichtige Wörter normalerweise seltener vorkommen als unwichtige, wie z.B. Funktionswörter oder Adverbien. Der Beispielsatzkorpus, der für die Übersetzung dieser Wörter zur Verfügung steht, wird dadurch nicht unnötig eingeschränkt. Die Sortierung selbst ist durch die Verwendung einer Index-Datei, in der zu jedem Wort die vorhandenen Beispielsatzpaare abgespeichert sind, sehr effizient möglich.

Die Liste aufsteigend nach der Anzahl der Beispielsatzpaare zu sortieren, bringt noch einen zweiten Vorteil:

Einige Wörter kommen extrem häufig vor, z.B. enthält jeder vierte Satz in dem verwendeten Beispielkorpus *ja* bzw. *ich*. Deshalb muß die Anzahl der zu betrachtenden Beispielsatzpaare aus Performancegründen beschränkt werden, z.Zt. auf 100 Beispielsatzpaare. Um eine möglichst geringe Anzahl relevanter Beispielsatzpaare auszusortieren, werden auf jeden Fall sämtliche bislang für diesen Satz betrachtete Beispielsatzpaare weiter verwendet.

Anhand eines Beispiels soll dies verdeutlicht werden. Wenn der zu übersetzende Satz

Ich finde Ende Januar eigentlich ganz gut

lautet und im Beispielkorpus das Satzpaar

ich finde <T> gut →
<T> would be fine for me

vorhanden ist, wäre es fatal, wenn zuerst *ich* übersetzt wird und das oben genannte Beispielsatzpaar aus Performancegründen vorab aussortiert wird. Ist dies das einzige Beispielsatzpaar mit *finden*, kann der Satz womöglich nicht mehr korrekt übersetzt werden. Je nach Alignment wird entweder das Beispielsatzpaar gar nicht mehr gefunden (*ich* und *finde* zusammengelinkt) oder *ich* wird fälschlicherweise mit *I* übersetzt (*ich* und *finde* ist nicht zusammengelinkt), weil das Satzpaar für die Übersetzung von *ich* nicht betrachtet wird. Durch die Sortierung wird erst das seltenere *finden* und dann *ich* übersetzt und beidesmal das richtige Beispielsatzpaar ausgewählt.

Falls für ein Wort überhaupt kein Beispielsatzpaar gefunden wurde, wird in zwei externen Wörterbüchern [Deu98a],[Deu98b] nach einer entsprechenden Übersetzung gesucht. Dabei wird allerdings kein Kontext mehr berücksichtigt, d.h. es findet eine reine Wort-zu-Wort-Übersetzung statt. Durch Einbinden der externen Wörterbücher können auch noch Sätze bearbeitet werden, die außerhalb des eigentlichen Domänenbereichs liegen. Diese werden wegen der fehlenden Berücksichtigung des Kontextes zwar eventuell nicht mehr so treffend übersetzt, können aber immerhin überhaupt übersetzt werden.

Nachdem nun für die ausgangssprachlichen Wörter die entsprechenden zielsprachliche Übersetzungen mit Hilfe der Auswahlfunktion gefunden sind, werden die

Übersetzungen in den zielsprachlichen Syntaxbaum eingefügt. Die syntaktische Struktur wird dabei aus dem zielsprachlichen Teil des Beispielsatzes so weit wie möglich übernommen, der Rest stammt aus dem zu übersetzenden ausgangssprachlichen Satz. Der zielsprachliche Syntaxbaum wird auf diese Art schrittweise erweitert.

Abschließend soll noch kurz dargestellt werden, warum eine einfache Top-Down-Vorgehensweise, wie sie im ATR-System eingesetzt wird und bei der Knoten für Knoten einzeln übersetzt werden, hier nicht eingesetzt werden kann:

Im ATR-System arbeitet die syntaktische Analyse und das Transfermodul Hand in Hand: So erzeugt die syntaktische Analyse einen Baum, der nur aus Templates besteht, für die entsprechende Translation Templates zur Verfügung stehen. Diese ausgangssprachlichen Templates können direkt eins zu eins mit Hilfe der Translation-Template-Paare in dem Syntaxbaum durch zielsprachliche Templates ersetzt werden. Diese bilden dann den zielsprachlichen Syntaxbaum. Diese Vorgehensweise ist bei meinem System nicht möglich, da einerseits die Translation Templates mehrere Knoten beinhalten können, andererseits ein Knoten im zielsprachlichen Synatxbaum aus mehreren Translation Templates zusammengesetzt sein kann (s.a. Abb. 6.1).

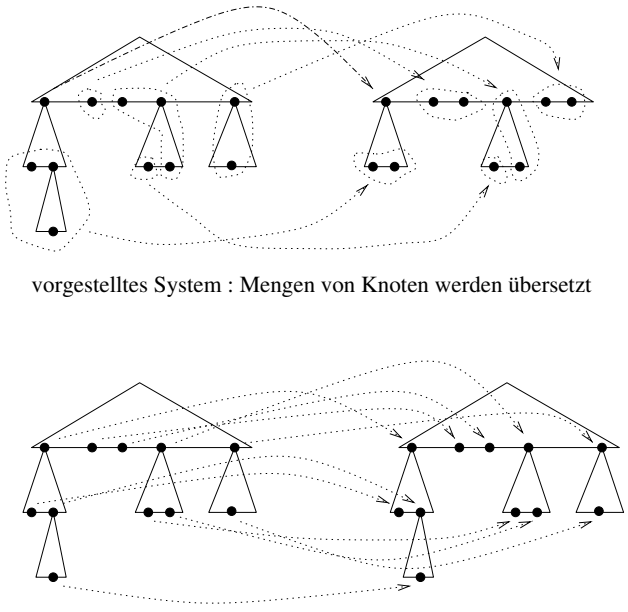

Abbildung 6.1: Vergleich Transfer bei ATR und meinem System

Die Hauptfunktion `translate` ist nachfolgend noch als Pseudocode zusammen mit Erläuterungen dargestellt, wobei die oben genannte Vorgehensweise angewandt wird.

```
( 1)  proc translate(g)
( 2)    wordlist = g.getWordlist()
( 3)    pairlist = []
( 4)    e = new chunk
( 5)    // Auswahl der besten Templates
( 6)    sort(wordlist)
( 7)    while wordlist.isNotEmpty() do
( 8)      pair = findBestTranslation(wordlist.first())
( 9)      wordlist.remove(pair.getSourceWords())
(10)      pairlist.add(pair)
(11)    end
(12)    // zielsprachlichen Syntaxbaum aufbauen
(13)    sort(pairlist)
(14)    foreach pair in pairlist do
(15)      insertInTarget(pair,e)
(16)    end
(17)    sortWords(e) // Generierung
(18)    return e
(19)  end
```

Der Funktion `translate` wird der ausgangssprachliche Syntaxbaum übergeben und als Resultat liefert sie einen zielsprachlichen Syntaxbaum zurück. Die Funktion `getWordlist` (2) schreibt alle vorkommenden Wörter in eine Liste. Diese Liste wird aufsteigend nach der Häufigkeit der für das jeweilige Wort vorhandenen Beispielsatzpaare sortiert (6). Den passendsten Beispielsatz sucht die Funktion `findBestTranslation` (8) für das erste Wort von `wordlist` heraus (siehe Abschnitt 6.2). In (9) werden alle Wörter, die übersetzt wurden, aus `wordlist` gelöscht. Zudem werden die ausgewählten Beispielsatzpaare in einer Liste gespeichert (10). Diese Liste wird so sortiert, daß die Reihenfolge der Paare der Reihenfolge der Wörter im zu übersetzenden Satz entspricht (13). Der Aufruf der Funktion `insertInTarget` (15) fügt nun die entsprechenden Teile des herausgesuchten Beispielsatzpaares in den zielsprachlichen Baum ein (siehe Abschnitt 6.4). Die Sortierung in (13) sorgt dafür, daß die ausgangssprachliche Wortfolge weitgehend im zielsprachlichen Syntaxbaum erhalten bleibt. Der Aufruf der Funktion `sortWords` (17) sortiert die Wörter im zielsprachlichen Syntaxbaum `e` so, daß der natürlich-sprachliche Satz ausgegeben werden kann, indem der Syntaxbaum in infix-Ordnung durchlaufen wird.

6.2 Auswahl der Beispielsätze

Überlegungen zur Vorgehensweise

Meist existieren für ein Wort mehrere verschiedene Übersetzungsmöglichkeiten. Deshalb muß in den Beispielsatzpaaren das geeignetste Beispielsatzpaar gefunden werden. Diese Aufgabe übernimmt die Auswahlfunktion, wobei alle Beispielsatzpaare, die das zu übersetzende Wort enthalten, der Reihe nach durchgegangen und bewertet werden. Der am höchsten bewertete Beispielsatz wird für die Übersetzung verwendet. Für die Bewertung ist eine Distanzfunktion notwendig, die überprüft, wie gut der jeweilige Beispielsatz zu dem zu übersetzenden Satz paßt. Die in meinem System implementierte Distanzfunktion berücksichtigt bis dato nur den gerade zu übersetzenden Satz, d.h. weder die Sätze davor noch danach, und auch noch keine externen Informationen, die z.B. von einer semantischen Komponente geliefert werden könnten. Durch die mengenorientierte Verarbeitung können solche Informationen relativ problemlos in das System integriert werden. So könnten bei meinem System z.B. noch von dem Dialogmodul des VERBMOBIL-Systems ermittelten Dialogakte zur Desambiguierung eingesetzt werden.

Die Bewertungsfunktion verwendet bei meinem System im Unterschied zum ATR-System den vollständigen ausgangssprachlichen Beispielsatz und nicht nur einzelne Wörter. Die Distanzfunktion arbeitet auch stets nach dem gleichen Schema und ist im Gegensatz zum ATR-System nicht abhängig davon, ob ein innerer Knoten oder ein Blatt im Baum übersetzt wird.

Die Distanzfunktion hat zwei Aufgaben zu erfüllen:

1. Eine möglichst weitgehende Zuordnung der Wörter bzw. Knoten aus dem zu übersetzenden Satz und dem ausgangssprachlichen Teil des Beispielsatzpaares.
   Dies geschieht mit Hilfe einer Distanzfunktion, die zwei Wörter vergleichen und auf ihre Ähnlichkeit prüfen kann. Dazu kann wie beim ATR-System eine semantische Worthierarchie verwendet werden. Wegen des hohen Implementierungsaufwandes wurde dieser Ansatz noch nicht realisiert und es wird nur auf die Übereinstimmung der Schreibweise geprüft. Bei einigen Wortarten werden zusätzlich noch die Stammformen beider Wörter berücksichtigt. Wörter bzw. Knoten im zu übersetzenden Satz und im ausgangssprachlichen Beispielsatz, für die keine Entsprechung gefunden werden konnte, berücksichtigt die Auswahlfunktion bei der Bewertung entsprechend.

2. Die ermittelten Wortdistanzen müssen noch zu einem singulären Wert verrechnet werden. Dazu werden die einzelnen Werte gewichtet. Das Gewicht ist u.a. abhängig von der POS des bewerteten Wortes und des Wortes, das übersetzt werden soll. Die Verwendung von Gewichten ist für die richtige Auswahl der Beispielsatzpaare unverzichtbar. Anfängliche Tests ohne Gewichtung haben gezeigt, daß relativ häufig nicht geeignete Beispielsätze ausgewählt wurden.

Zuordnung der Wörter bzw. Knoten

Der Algorithmus zur Zuordnung der Wörter wird hier vorgestellt, anschließend werden die einzelnen Funktionen näher beschrieben.

```
( 1)  proc match(word,s,p)
( 2)    <sc,pc> = findTop(word,s,p)
( 3)    return matchChunk(sc,pc)
( 4)  end

( 5)  proc matchChunk(s,p)
( 6)    foreach token st of s do
( 7)      foreach token pt of p do
( 8)        if st.isWord() and pt.isWord() then
( 9)          table[st,pt] = matchWord(st,pt)
(10)        else if st.isChunk() and pt.isChunk() and
(11)                st.name()==pt.name() then
(12)          table[st,pt] = matchChunk(st,pt)
(13)        else
(14)          table[st,pt] = nullAssessment
(15)        end
(16)      end
(17)    end
(18)    while not table.isEmpty() do
(19)      table.findBestAndDelete()
(20)    end
(21)    return computeValueWithWeights(table)
(22)  end
```

Da die Syntaxbäume des Beispielsatzes und des Quellsatzes unterschiedlich hoch sein können, muß zuerst der Teil in beiden Bäumen ermittelt werden, der auf Übereinstimmung geprüft werden kann. Dazu dient der Aufruf der Funktion `findTop` (2). Diese Funktion geht von dem Wort aus, das übersetzt werden soll und läuft solange in beiden Bäumen parallel hoch, bis bei einem der Bäume die Spitze erreicht ist oder die Knoten unterschiedliche Bezeichnungen haben. Diese ermittelten Stellen bilden dann den Ausgangspunkt für den weiteren Vergleich der Bäume. Steht z.B. das zu übersetzende Wort in einem Nebensatz, der Beispielsatz hat aber keinen Nebensatz, wird der Hauptsatz des zu übersetzenden Satzes nicht mit in die Bewertung einbezogen (siehe Abb. 6.2)

Die Funktion `matchChunk` versucht nun für zwei gegebene Knoten `s` (für source) und `p` (für pattern) herauszufinden, wie gut sie übereinstimmen und gibt einen singulären Wert als Resultat zurück. Dazu vergleicht sie zuerst jedes Token (also Wort oder Knoten) in `s` mit jedem Token in `p`. Zum Vergleichen von zwei Wörtern wird die Wortdistanzfunktion `matchWord` (9) verwendet. Falls `s` und `p` Knoten mit gleicher Bezeichnung (also z.B. zwei NP-Knoten) sind, wird `matchChunk` (10-12) rekursiv aufgerufen, um herauszufinden, inwieweit die beiden Knoten übereinstimmen. Für die restlichen Fälle, d.h. wenn `s` und `p` aus einem Wort und einem

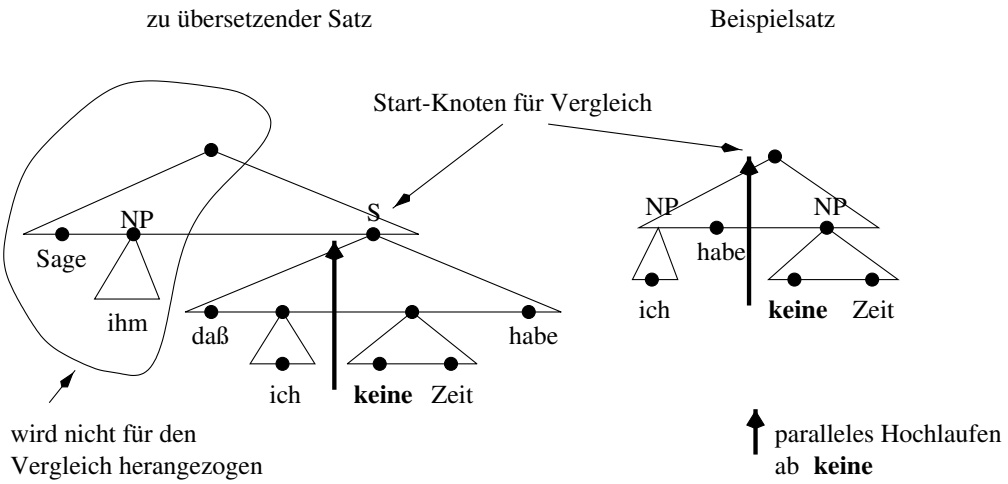

Abbildung 6.2: Baumvergleich zur Übersetzung von *keine*

Knoten bestehen oder beide Knoten unterschiedliche Bezeichnungen haben, wird der Wert *keine Übereinstimmung* eingetragen (14). Das Ergebnis der Funktion `matchWord` bzw. dem rekursiven Aufruf von `matchChunk` wird in einer Matrix abgespeichert.

Anhand des zu übersetzenden Satzes

> *[S Wahrscheinlich hast [NP-1 du] [PP.T am Donnerstag] [NP-2 keine Zeit]]*

und des im Beispielkorpus enthaltenen Satzes

> *[S ja [PP.T <T>] habe [NP-1 ich] [NP-2 keine Zeit]]*

soll die Funktionsweise der Zuordnung der Wörter konkret aufgezeigt werden.

Als Beispiel ist die Matrix beim Aufruf von `matchChunk` mit den beiden S-Knoten dargestellt. In der Horizontalen sind die Wörter aus dem Beispielsatz aufgelistet, in der Vertikalen die Wörter aus dem zu übersetzenden Satz. Die Matrizen für die einzelnen rekursiven Aufrufe von `matchChunk` mit den *NP-*, *PP-* und *PP.T-*Knoten sind nicht extra aufgelistet, aber in der Tabelle mit einem * markiert. Die in jeder Zeile bzw. Spalte ausgewählten Werte sind in der Tabelle fett hervorgehoben. Darauf, wie die Werte im einzelnen zustande kommen, wird später noch eingegangen.

	PP.T	habe	NP-1	NP-2
wahrscheinlich	-1	-1	-1	-1
hast	-1	**0.8**	-1	-1
NP-1	-1	-1	**-1** *	-2 *
PP.T	**0.8** *	-1	-1	-1
NP-2	-1	-1	-1 *	**2** *

Aus der so gewonnenen Matrix muß nun ein singulärer Wert ermittelt werden. Mehrere Möglichkeiten bieten sich dabei an. Zum einen kann einfach der Mittelwert der gesamten Matrix genommen werden. Dies hat allerdings den Nachteil, daß längere Sätze benachteiligt werden, da die Anzahl der nicht übereinstimmenden und damit negativ bewerteten Token mit zunehmender Länge stärker wächst als die der übereinstimmenden Token.

Ein anderer Ansatz, der auch bei meinem System verwendet wird, ist, die Token aus `p` und `s` soweit wie möglich paarweise zuzuordnen. Die Bewertung für die Paare wird zusammen mit den restlichen, nicht in Paaren vorkommenden Wörtern aus `p` und `s` gewichtet verrechnet. Die Zuordnung der Wörter sollte so gewählt werden, daß die Summe der Bewertungen möglichst hoch wird. In meinem System wird dazu in der Matrix mit der Funktion `findBestAndDelete` (19) nach der höchsten Bewertung gesucht. Falls mehrere höchste Bewertungen existieren, wird die zuerst gefundene genommen. Dann wird die Zeile und die Spalte, die die Bewertung enthält, aus der Tabelle gestrichen. Somit wird kein Wort zweimal mit einem anderen Wort gekoppelt. `findBestAndDelete` wird solange wiederholt aufgerufen, bis die Tabelle leer ist (18-19). Dieses Vorgehen ermittelt zwar nicht die optimale Lösung bezüglich der Summe der Bewertungen, arbeitet jedoch effektiv und liefert im praktischen Einsatz gute Ergebnisse.[1]

In dem Beispiel werden also folgende Zuordnungen vorgenommen (mit angegeben sind die einzelnen Ergebnisse):

zu übersetzender Satz	Beispielsatz	Ergebnis
[NP-1 du]	[NP-1 ich]	0.8
[NP-2 keine Zeit]	[NP-2 keine Zeit]	2.0
hast	habe	0.8
[PP.T Donnerstag]	[PP.T <T>]	1.0

Für *wahrscheinlich* ist keine Zuordung vorhanden und wird somit mit -1.0 bewertet.

Diese Werte addiert die Funktition `computeValueWithWeights` (21) gewichtet auf.

Berechnung der Wortdistanz

Die Werte liegen in einem zu Null symmetrischen Intervall von [-1.0,1.0]. Der Vorteil, ein zu Null symmetrisches Intervall zu wählen, liegt darin, daß Wörter sowohl einen positiven als auch einen negativen Einfluß haben können, je nachdem ob sie übereinstimmen oder nicht. Alternative Experimente mit dem zu Null asymetrischen Intervall $[-1.0\ 0.0]$ zeigten, daß kurze Sätze zu sehr bevorzugt werden, da zusätzliche Wörter das Ergebnis nicht verbessern können. Umgekehrt würde ein Intervall $[0.0\ -1.0]$ längere Sätze bevorzugen, was allerdings nicht experimentell verifiziert wurde.

[1] Das Problem entspricht dem optimalen Zuweisungsproblem bei bipartiten Graphen.

Falls die Schreibweise der beiden Wörter übereinstimmt, wird die bestmögliche Bewertung 1.0 zurückgeliefert. Die POS wird dabei meist nicht berücksichtigt, um nicht durch Fehler des Taggers unnötig schlechte Resultate zu bekommen. Lediglich in Fällen, bei denen die POS eine entscheidende Rolle spielt, z.B. bei dem Wort *das*, das entweder Artikel oder Pronomen sein kann und deshalb unterschiedlich zu übersetzen ist, muß auch die POS übereinstimmen.

Stimmen lediglich die Stammformen der beiden Wörter überein, wird eine geringe Abwertung vorgenommen und nur ein Wert von 0.8 zurückgeliefert. Damit wird die Bevorzugung von Sätzen erreicht, bei denen die Schreibweise der Wörter exakt übereinstimmt.

Wenn nur die POS der beiden Wörter übereinstimmt, wird die negativste Bewertung mit -1.0 genommen, derselbe Wert, als würde die POS nicht übereinstimmen. Allerdings wird in dem Rückgabeobjekt vermerkt, daß die POS übereingestimmt haben. Dies spielt später bei der Gewichtung eine Rolle.

Gewichtung

Überlegungen zur Vorgehensweise

Bei der Gewichtung bestehen prinzipiell zwei Möglichkeiten:
Lokale und globale Gewichtung.

Bei der *lokalen Gewichtung* ist jedem Link-Wort-Paar in allen Beispielsatzpaaren ein Gewicht zugeordnet, d.h. jede Gruppe von Wörtern, die gemeinsam übersetzt werden, erhält für jedes Wort im Satz eigene Gewichte. Anhand von Trainingsdaten werden diese Gewichte für jeden Beispielsatz ermittelt. In [FI92a] findet sich ein Vorschlag, wie diese Gewichte bestimmt werden können. Dabei wird bei Beispielsätzen mit gleichen Übersetzungen überprüft, wie relevant die einzelnen Wörter für das Übersetzungsergebnis sind. Wegen der eingeschränkten Verfügbarkeit von Trainingsdaten (die vorhandenen Satzpaare sind wenig geeignet, da nur wenig redundant) und dem relativ hohen Implementierungsaufwand wurde bei meinem System dieser Weg nicht eingeschlagen. Eine nachträgliche Erweiterung dieses Systems um eine entsprechende Komponente ist allerdings relativ problemlos möglich. Die Gewichte können als Attribute den einzelnen Wörtern in den Beispielsätzen einfach zugefügt werden.

Bei der *globalen Gewichtung* hängt das Gewicht eines jeden Wortes von dem Wort selbst und dem gerade zu übersetzenden Wort ab und ist nicht vom konkreten Beispielsatz abhängig. Zudem kann berücksichtigt werden, wie groß der Abstand zwischen dem zu gewichtenden Wort und dem zu übersetzenden Wort ist, d.h. wieviele Knoten sich zwischen beiden Wörtern befinden. Natürlich können nicht bei allen Wörtern für sämtliche Abstände stets einzelne Werte angegeben werden, sondern sie müssen teilweise zusammengefaßt werden. Hier bieten sich z.B. die POS der Wörter an. Eine Annäherung an die lokale Gewichtung stellt eine Einteilung der Sätze in Klassen dar, wobei für jede Klasse eigene Werte vergeben werden können. So ist z.B. vorstellbar, daß bei feststehenden Ausdrücken Abweichungen zu größeren Abwertungen führen als im Normalfall.

Der Vorteil dieses Ansatzes liegt darin, daß bedingt durch die geringere Anzahl von Gewichten, die Gewichte manuell ohne Trainingsdaten eingestellt werden können. Deshalb wurde dieser Ansatz implementiert.

Implementierung

Nachdem die Funktion `matchChunk` die Wörter soweit möglich zugeordnet und bewertet hat, werden die Gewichte bestimmt. Die einzelnen Bewertungen werden mit den Gewichten multipliziert und aufaddiert. Die betrachteten Wörter sind die Wörter aus dem Beispielsatz plus allen nicht zugeordneten Wörtern.

Sämtliche Wörter im Beispielsatz und die restlichen Wörter im zu übersetzenden Satz, die in der Funktion `matchChunk` keinem Wort im Beispielsatz zugeordnet werden konnten, werden mit Gewichten versehen und mit den gefundenen Bewertungen multipliziert.

Um die Gewichte für jedes Token zu finden, wird für jedes Token eine Liste mit folgenden Bedingungen erstellt:

1. Das zu übersetzende Wort (nachfolgend als `w1` bezeichnet)
2. Das Wort bzw. der Knoten selbst, für den das Gewicht ermittelt wird (nachfolgend als `w2` bezeichnet)
3. Der Typ der Übereinstimmung von `w2` und dem entsprechenden Wort im Beispielsatz, der breits von `matchChunk` bzw. `matchWord` bestimmt wurde. Mögliche Werte sind dabei :

   full Übereinstimmung der Schreibweise bei Wörtern bzw. der Bezeichnung bei Token

   stemm Übereinstimmung bei den Stammformen

   pos Übereinstimmung bei der POS der Wörter

   missingS keine Entsprechung von `w2` im zu übersetzenden Satz

   missingP Wort im zu übersetzenden Satz hat keine Entsprechung im Beispielsatz

4. Der Pfad zwischen `w1` und `w2`, d.h. welche Knoten durchlaufen werden müssen, um von `w1` nach `w2` zu gelangen. Beispiele für Pfade sind: -NP+PP, -S+NP (- steht für eine Ebene hoch gehen, + für Ebene tiefer gehen), = (zwischen `w1` und `w2` liegt kein Knoten)

Aufgrund dieser Liste wird nun in einer Tabelle (siehe Tab. 6.1) nach dem passenden Gewicht gesucht, indem die Tabelle von oben nach unten durchlaufen wird. In den einzelnen Zeilen der Tabelle stehen für jedes Element in der Liste Bedingungen und das zugehörige Gewicht. Sind alle Bedingungen erfüllt, wird das Gewicht aus dieser Zeile genommen. Die Bedingungen sind ähnlich implementiert wie Bedingungen an Übergängen in den Automaten. An die Bedingungen

sind Funktionen gekoppelt, die den entsprechenden Test ausführen. Das hat den Vorteil, daß die Bedingungen recht flexibel ausgelegt werden können und sich somit der Umfang der Tabelle erheblich reduzieren läßt. So sind z.B. Funktionen vorhanden, die die POS der Wörter testen und die zusätzlich auch mit POS-Klassen umgehen können (z.B. alle Verben). Zusätzlich existiert eine Funktion *, die immer wahr ist.

	w1	**w2**	**Typ**	**Pfad**	**Gewicht**
1	VV	NP	missingS	=	3.0
2	haben	VVPP	missingP	=	5.0
3	V	NP	missingP	=	3.0
4	V	PP	missingP	=	1.5
5	VFIN	VVINF	missingP	=	10.0
6	PTKNEG	VFIN	missingP	=	5.0
7	ART	NN	missingP	=	5.0
8	PDS	NN	missingP	=	5.0
9	PIS	NN	missingP	=	5.0
10	PIS	VFIN	missingP	-NP	5.0
11	PPER	VFIN	missingP	-NP	5.0
12	ADJD	VV	missingP	=	5.0
13	ADV	VV	missingP	=	3.0
14	PTKANT	VV	missingP	=	3.0
15	*	*	missingP	*	0.0
16	PDS	VV	cite	-NP	3.0
17	PDS	VV	stamm	-NP	3.0
18	sein	ADJD	*	=	3.0
19	PTKA	ADJD	cite	=	4.0
20	APP	VV	*	-PP	1.0
21	APP	*	*	-PP	0.2
22	PPER	VV	*	-NP	3.0
23	PRF	VV	*	-NP	3.0
24	*	*	*	S	0.2
25	*	ADV	*	*	0.3
26	*	ITJ	*	*	0.3
27	*	PTKANT	*	*	0.3
28	*	ART	*	*	0.3
29	*	*	*	*	1.0

Tabelle 6.1: Gewichte mit zugehörigen Bedingungen

Da es bei meinem System möglich ist, gleich mehrere Wörter zusammen zu übersetzen und außerdem das Gewicht vom zu übersetzenden Wort abhängig ist, wird für jedes zu übersetzende Wort das Gewicht bestimmt. Abschließend wird der Mittelwert der Gewichte berechnet. Denkbar wäre auch eine Gewichtung je nach Wertigkeit der einzelnen zu übersetzenden Wörter. Dieser Ansatz wurde aber als nicht so relevant eingestuft und deshalb nicht implementiert.

Nun soll noch auf den Zweck der einzelnen Zeilen in Tabelle 5.1 eingegangen werden, wobei als Standardgewicht 1.0 (Z.29) ist.

Die Anzahl der NPs bei Verben ist, wie sich herausgestellt hat, sehr wichtig für

die Übersetzung von Verben und wird deshalb entsprechend berücksichtigt. (Z.1) sorgt für Abwertung, wenn im ausgangssprachlichen Satz weniger NPs vorhanden sind, (Z.3) wenn im Beispielsatz weniger NPs vorhanden sind.

Beispielsatzpaare mit abweichender PP sollen nicht negativer bewertet werden als Beispielsatzpaare mit fehlender PP (Z.4).

Normalerweise stören zusätzlich vorhandene Wörter im zu übersetzenden Satz kaum und führen deshalb nicht zur Abwertung (Z.15). Ausnahmen hiervon sind in (Z.2-Z.14) aufgeführt. So hängt z.B. bei einer Reihe von Wörtern die Übersetzung stark davon, ob ein Verb im Satz vorkommt oder nicht. In letzterem Fall handelt es sich häufig um Ellipsen, die zum Teil recht frei übersetzt werden. Betroffene Wortgruppen sind vor allem Adjektive (Z.12) und Adverbien (Z.13). Außerdem irrt sich der POS-Tagger ab und zu beim Unterscheiden von Demonstrativpronomen und Artikeln, die im zielsprachlichen völlig unterschiedlich übersetzt werden. Deshalb muß berücksichtigt werden, ob ein Nomen (NN) innerhalb des NP-Knotens vorhanden ist (Z.7-Z.9). Als Sonderfälle zählen auch die unterschiedlichen Anwendungen von *haben* als Voll- oder Hilfsverb, was ebenfalls der POS-Tagger nicht immer korrekt unterscheiden kann (Z.2). Bei der Übersetzung von *nicht* muß ebenfalls unterschieden werden, ob ein Verb vorhanden ist oder nicht (Z.6).

Die Übersetzung von Wörtern mit der Stammform *sein* hängt stark vom Adjektiv im Satz ab, deshalb werden sie in diesem Fall höher gewichtet (Z.18). Für Präpositionen hingegen ist das Verb relativ wichtig (Z.20), die restlichen Wörter im Satz werden geringer gewichtet als alles, was innerhalb der Präposition steht (Z.21).

Wörter in Nebensätzen sind ebenfalls nicht so relevant (Z.24), genauso wie Adverbien und ähnliche Wortgruppen (Z.25-Z.28).

6.3 Verwendung eines zweisprachigen Wörterbuchs

Falls bei der Übersetzung eines Wortes kein Beispielsatzpaar gefunden wird, wird auf zwei bilinguale Wörterbücher zurückgeriffen mit 126000 und 27000 Einträgen [Deu98a],[Deu98b]. Diese beiden Wörterbücher sind frei im Internet verfügbar. Es handelt sich in den Wörterbüchern um reine Wort-zu-Wort-Übersetzungen, kein weiterer Kontext ist angegeben. Die ausgangssprachliche POS wird mittels einer Tabelle, ähnlich der Tab. 5.1, in eine zielsprachliche POS umgewandelt, da die Generierungskomponente zum Sortieren auf eine den Wörtern zugeordnete POS angewiesen ist. Einen Sonderfall stellt die Übersetzung von Verben dar. Damit die NPs und PPs von der Generierungskomponente in die richtige Reihenfolge gebracht werden können, wird nicht nur das Wortpaar weitergegeben, sondern es werden zusätzlich noch NP- und PP-Knoten erzeugt, die bei der Bewertung mit entsprechenden Knoten des Ausgangssatzes verknüpft werden. In diesem Fall muß allerdings davon ausgegangen werden, daß das Subjekt vor den Objekten steht,

da keine Kontextinformation verfügbar ist und keine morphologische Analyse stattfindet.

6.4 Einbau in den zielsprachlichen Syntaxbaum

Nachdem nun ein passendes Beispielsatzpaar ausgewählt wurde, wird die Übersetzung in den Zielbaum eingefügt. Da nicht nur Wörter eingefügt werden, sondern parallel auch eine syntaktische Struktur aufgebaut werden muß, gestaltet sich der Einbau komplizierter. Die Sortierung der Wörter innerhalb eines Knotens wird erst nachträglich von der Generierungskomponente vorgenommen.

Prinzipiell wird beim Einbau wortweise vorgegangen. Allerdings muß im Zielsatz für die Generierung auch eine syntaktische Struktur entstehen. Deshalb können die Wörter nicht einfach kopiert werden. Es genügt auch nicht, aus dem zielsprachlichen Beispielsatz die Struktur nur zu übernehmen, da dort u.U. die vollständige Struktur nicht vorhanden ist. Z.B. kann das zu übersetzende Wort in einem Nebensatz stehen, wenn der zur Übersetzung ausgewählte Beispielsatz jedoch gar keinen Nebensatz enthält, wird in einem solchen Fall die syntaktische Struktur für den Nebensatz (also der S-Knoten) aus dem Ausgangssatz übernommen. Der Aufbau findet also in zwei Schritten statt. Soweit notwendig, wird die Struktur aus dem Ausgangssatz übernommen und anschließend die Struktur des zielsprachlichen Teils des Beispielsatzpaares eingebaut. In Abb. 6.3 ist dies anhand eines Beispiels veranschaulicht.

Entsprechend sieht der Pseudo-Code aus:

```
(1)  proc addInTarget()
(2)    <sc,tp> = findLinkedChunk(pair)
(3)    pos = copyChunk(NULL,sc,target)
(4)    forall words w of pair.getTargetTokens() do
(5)      tpos = copyChunk(tp,w.mother(),pos)
(6)      copyToken(w,tpos)
(7)    end
(8)  end
```

Um zu erkennen, welcher Teil aus dem zu übersetzenden Satz und welcher Teil aus dem Beispielsatz übernommen werden soll, wird zuerst `findLinkedChunk` (2) aufgerufen. Diese Funktion sucht in dem ausgangssprachlichen Teil des Beispielsatzes vom zu übersetzenden Wort baumaufwärts nach dem ersten Knoten, der mit einem zielsprachlichen Knoten gekoppelt ist. Kann kein entsprechender Knoten gefunden werden, wird der Top-Knoten zurückgeliefert. Ab diesem Knoten aufwärts, allerdings ohne den Knoten selbst, wird die Struktur des Beispielsatzes kopiert (4-6). Der Rest wird aus dem Ausgangssatz kopiert (3). Die beiden Knoten, die die Funktion `findLinkedChunk` als Ergebnis zurückliefert, sind folglich der Knoten im Ausgangssatz, bis zu dem kopiert wird und der Knoten im zielsprachlichen Teil, ab dem kopiert wird.

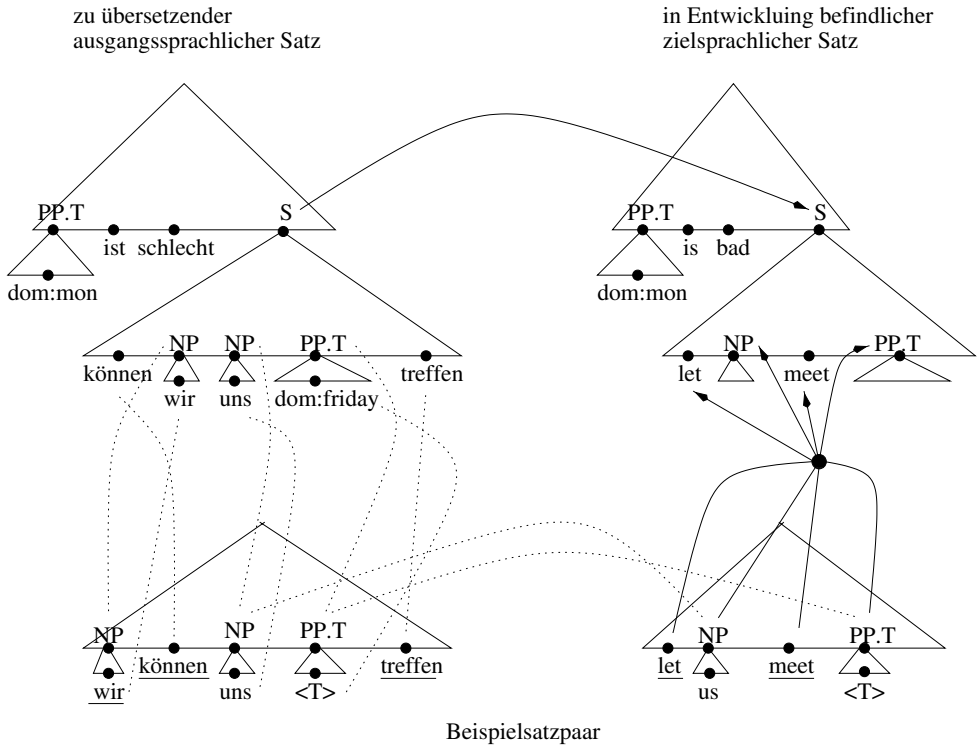

Abbildung 6.3: Einfügen von *let* [*NP*] *meet* [*PP.T*]. *Us* und *dom:fri* wird im nächsten Schritt eingefügt, da es separat gelinkt ist. Die Pfeile beschreiben den Kopiervorgang, die gestrichelten Linen Zuordnungen (zwischen den ausgangssprachlichen Sätzen) und Links (zwischen dem ausgangssprachlichen und dem zielsprachlichen Beispielsatz)

Weshalb prinzipiell nicht der Top-Knoten, sondern ein eventuell vorhandener gekoppelter Knoten genommen wird, hat folgende Ursache: Gekoppelte Knoten können als unabhängige Einheiten betrachtet werden, deshalb müssen die darüberliegenden Knoten nicht übereinstimmen. Anhand eines Beispiels soll das näher erläutert werden.

Wenn das Wort *Konferenzzimmer* in dem Satz

[*NP Das Konferzzimmer*] *ist belegt*

übersetzt werden soll, aber das Wort im ausgesuchten Beispielsatzpaar innerhalb einer PP vorkommt, wie in

[*S Treffen* [*NP wir*] [*NP uns*] [*PP in* [*NP dem Konferenzzimmer*]] →
[*S Let* [*NP us*] *meet* [*PP in* [*NP the conference room*]]

darf der PP-Knoten natürlich nicht mitkopiert werden. Deshalb wird erst der Teilbaum oberhalb dieses Knotens aus dem Beispielsatz kopiert.

Die Funktion `copyChunk` kopiert die entsprechenden Teilbäume, wobei das erste Argument den Startknoten (exklusiv), das zweite den Endknoten (inklusive) und das dritte den Zielknoten angibt. Beim Kopieren wird zuerst gesucht, ob bereits Knoten in früheren Durchläufen kopiert wurden. Nur wenn dies nicht der Fall ist, werden tatsächlich die Knoten kopiert.

Die Funktion `copyToken` schließlich kopiert das Wort bzw. den Knoten in den Zielbaum. Bei einem Knoten wird zusätzlich noch überprüft, ob er bereits früher kopiert wurde. Der Knoten wird in diesem Fall als dem Beispielpaar zugehörig markiert. Dies ist in der Generierung später Voraussetzung, um die Knoten in der richtigen Reihenfolge anordnen zu können.

In der Hauptfunktion `addInTarget` kann die Funktion `findLinkedChunk` und der 1.Aufruf von `copyChunk` (2-3) aus der Schleife, die über die einzufügenden zielsprachlichen Wörter läuft, herausgenommen werden, da beide Funktionen unabhängig von `w` sind.

6.5 Generierung

Bevor der Syntaxbaum linearisiert, d.h. als normaler zielsprachlicher Satz ausgegeben werden kann, müssen innerhalb jeden Knotens die einzelnen Wörter und Unterknoten in die richtige Reihenfolge gebracht werden. Dies liegt daran, daß die Transferkomponente die Token innerhalb eines Knotens als Menge auffaßt und somit die Reihenfolge nicht berücksichtigt wird.[2] Bewegungen über Knotengrenzen finden in der Generierung nicht mehr statt. Diese Aufgabe übernimmt vollständig die Transferkomponente. Eine zweite Aufgabe der Generierungskomponente ist die morphologische Anpassung der Wörter. Wegen des hohen Implementierungsaufwands wurde bei meinem System darauf verzichtet.

Die hier implementierte Generierung arbeitet nach folgendem Prinzip:

Die Reihenfolge für die einzelnen Wörter wird – so weit wie möglich – aus den Beispielsätzen übernommen. Bei den Wörtern, bei denen dies mit dem verwendeten Verfahren nicht möglich ist, wird auf einen hierarchischen Automaten zurückgegriffen, der einzelne Wörter einfügen kann. Falls auch dies nicht gelingt, wird die Wortstellung aus dem ausgangssprachlichen Satz adaptiert. Abschließend finden noch kleinere Nachbearbeitungen statt, im Wesentlichen um einen Wechsel von Frage- zu Aussagesatz zwischen Beispielsatz und zu übersetzenden Satz auszugleichen.

Die konkrete Umsetzung dieses Ansatzes erfolgt so, daß die Liste mit Token im Knoten nicht direkt umsortiert wird, sondern eine neue Liste Schritt für Schritt wieder zusammengesetzt wird. Als Ausgangspunkt, d.h. als erstes Wort, das in die leere Liste eingefügt wird, dient bei S-Knoten das Vollverb, bei NP-Knoten das Nomen und bei PP-Knoten die Präposition. Diese Wortarten werden als Ausgangspunkt genommen, da sie die zentralen Bestandteile des jeweiligen Knotens

[2] Die Transferkomponente übernimmt – wie geschildert – weitgehend die Reihenfolge aus dem ausgangssprachlichen Satz, lediglich bei der Übersetzung mehrerer Wörter gleichzeitig, stimmt sie nicht mehr überein.

sind und die meisten Unterknoten an sie gekoppelt sind. Alle Wörter und Knoten, die aus dem gleichen Beispielsatz stammen, werden in der gleichen Reihenfolge wie im Beispielsatz angeordnet. Damit wird erreicht, daß die Unterknoten gleich richtig plaziert werden, was in den nachfolgenden Schritten nicht mehr so leicht möglich wäre.

Danach werden die restlichen Wörter und Knoten, die aus anderen Beispielsätzen stammen, einzeln eingefügt. In drei Stufen wird nun versucht, die einzelnen Wörter einzufügen. Wörter, die in einer Stufe nicht verarbeitet werden konnten, werden an die nächste weitergereicht. Die letzte Stufe ist so ausgelegt, daß sie auf jeden Fall jedes Wort verarbeiten kann. Die einzelnen Stufen werden nachfolgend erklärt:

- Zuerst wird versucht, das Wort anhand seiner Umgebung im Beispielsatz einzufügen. Wenn die Umgebung, d.h. die POS des rechten und linken Nachbarwortes, die das Token in dem Beispielsatz hat, auch in der bislang aufgebauten zielsprachlichen Liste gefunden wird, wird das Wort an die entsprechende Stelle kopiert.

- Anschließend werden Automaten benutzt, um noch nicht verarbeitete Wörter einzufügen. Um die Automaten für diese Aufgabe verwenden zu können, wurden sie um einen Befehl erweitert. Dieser ermöglicht das Verwenden eines zweiten Eingabestroms. Die bislang aufgebaute Liste wird als erster Eingabestrom verwendet, das einzufügende Token als zweiter Eingabestrom, der somit nur aus einem einzigen Token besteht. Ist der Automat erfolgreich, wird das Ergebnis als neue zielsprachliche Liste genommen. Ein Beispiel für einen Automaten findet sich in Abb. 6.4.

- In der dritten Stufe wird versucht die Position aus der ursprünglichen Liste zu übernehmen, die aus der ausgangssprachlichen Wortstellung herrührt. Dazu wird ermittelt, ob sich das einzufügende Token vor sämtlichen Token befindet, die im allerersten Schritt eingefügt wurden. Falls dies der Fall ist, wird das Token am Anfang eingefügt, sonst am Ende.

Abschließend werden am komplett zusammengesetzten Satz noch ein paar Änderungen im Verbumfeld vorgenommen, die ebenfalls von hierarchischen Automaten vorgenommen werden. Im einzelnen handelt sich dabei um folgende Punkte:

- Ein eventuell vorhandenes *do* aus Fragesätzen bzw. negierten Sätzen muß entfernt werden, da sie normalerweise durch `alignRest` mit dem Verb beim Alignment gekoppelt wurden.

- Falls es sich bei dem zu übersetzenden Satz um einen Fragesatz bzw. einen negierten Satz handelt, muß ein *do* eingefügt werden, bzw. Hilfsverben entsprechend umgestellt werden.

- Mehrere hintereinander stehende Hilfsverben, die durch die Verwendung von Stammformen beim Heraussuchen aus den Beispielsatzpaaren verursacht werden, werden teilweise gelöscht.

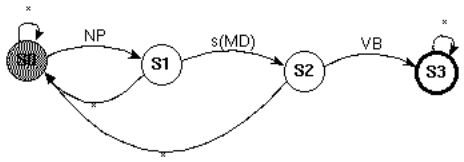

Abbildung 6.4: Ein Generierungsautomat, der Hilfsverben einfügt

Als Pseudocode sieht dies dann so aus:

```
( 1)  proc sort(e)
( 2)    forall chunks ec of e do
( 3)      sort(ec)
( 4)      sortChunk(ec)
( 5)    end
( 6)    endSort(e)
( 7)  end

( 8)  proc sortChunk(ec)
( 9)    exampleList = splitIntoDifferentExamples(ec)
(10)    TcChunk new = findBasicAndRemove(exampleList)
(11)    tokenList = exampleList.getAllTokens()
(12)    tokenList = insertUsingExample(tokenList,new)
(13)    tokenList = insertUsingAutomaton(tokenList,new)
(14)    tokenList = insertUsingPosition(tokenList,new)
(15)  end
```

Die Funktion `splitIntoDifferentExamples` (9) bildet Gruppen von Wörtern, die aus den gleichen Beispielsätzen stammen. Die Funktion `findBasicAndRemove` (10) sucht den Beispielsatz, der als Ausgangspunkt bei der Sortierung dient. Die Funktionen in (12)-(14) entsprechen den drei vorgestellten Stufen, die einzelne Wörter einfügen. Die Funktion `endSort` (6) ist noch für die erwähnte abschließende Umsortierung verantwortlich.

Beispiel

Der Ablauf bei der Generierung wird anhand eines Beispiels erläutert.

Übersetzt werden soll folgender Satz:

am Donnerstag habe ich leider wirklich keine Zeit

Die dabei verwendeten Beispielsätze sind:

ich habe keine Zeit [*PP.T <T>*] → *I am not free* [*PP.T <T>*]

ich habe leider <T> schon etwas vor → *sorry I am occupied <T>*

das ist bei mir wirklich schlecht → *that is really bad for me*

Das Ergebnis aus der Transferkomponente sieht so aus:

[*S* [*NP I*] *am not free* [*PP.T dom:thu*] *sorry really*]

Die Generierungskomponente baut nun in mehreren Stufen eine neue sortierte Liste auf. Rekursiv werden nun alle Knoten bearbeitet, wobei hier nur die Bearbeitung des S-Knotens gezeigt wird, da dies bei den anderen Knoten trivial ist.

Angefangen wird mit einer neuen, leeren Liste:

[ ]

Stufe 1

Da es sich um einen S Knoten handelt, wird das Verb als Ausgangswort mit den dazu gehörenden Worten aus dem Beispielsatz genommen und der Liste hinzugefügt.

[[*NP I*] *am not free* [*PP.T dom:thu*]]

Stufe 2

Das Umfeld, d.h. die POS der benachbarten Wörter, wird zur Einsortierung herangezogen. *Sorry* kann mit dieser Methode einsortiert werden, da im englischen Beispielsatz links nichts und rechts ein NP-Knoten steht. Die gleiche Umgebung ist gegeben, wenn *sorry* am Anfang der Liste einsortiert wird.

sorry [*NP I*] *am not free* [*PP.T dom:thu*]

Stufe 3

Einsortierung mit Hilfe des Automatens aus Abb. 6.4

sorry [*NP I*] *am really not free* [*PP.T dom:thu*]

Stufe 4: TEL-Ausdrücke werden umgewandelt

Mit Hilfe eines Tools, das die TEL-Ausdrücke in englische Sprache umwandelt, und Linearisierung des Syntaxbaums wird daraus dann:

sorry I am really not free on Thursday

Die Funktion `endSort` hat in diesem Beispiel nichts zu tun.

6.6 Ausblick

Bei der Auswahl der Beispielsätze kann noch mehr Kontext einbezogen werden, z.B. Dialogakte, Elemente von den Sätzen davor, semantische Informationen von anderen Komponenten etc. Dies ist einfach möglich, indem die entsprechenden Informationen in den zugehörigen Knoten eingeblendet werden und somit von der Auswahlfunktion berücksichtig werden können.

Um nicht unnötigt viel Kontext bei jeder Übersetzung in den Satz einbauen zu müssen und damit das System zu verlangsamen, ist denkbar, die entsprechenden Informationen erst dann anzufordern, wenn diese tatsächlich in einem Beispielsatz vorkommen.

Zur Verbesserung der Wortdistanzfunktion kann zusätzlich die semantische Datenbank von VERBMOBIL herangezogen werden. Damit könnte, ähnlich wie bei dem ATR-System, eine abgestufte Wortdistanz ermittelt werden.

Die Funktion, die zwei ausgangssprachliche Bäume vergleicht, sollte noch so erweitert werden, daß Knoten in den Syntaxbäumen vorkommen dürfen, die ausblendbar sind. Diese Maßnahme ist notwendig, um Konjunktionen zwischen NPs bzw. PPs durch eigene Knoten repräsentieren zu können und trotzdem ein korrektes Matching mit Syntaxbäumen zu ermöglichen, bei denen die entsprechenden NPs bzw. PPs ohne Konjunktionen auftreten.

Wichtig wäre auch die Einbeziehung des zielsprachlichen Teils des Beispielsatzes, da die zielsprachlichen Teilbäume nicht immer problemlos miteinander gekoppelt bzw. nicht problemlos umgeformt werden können, so kann z.B. *let us do sth.* nicht in einer Frage verwendet werden.

Ein Ziel für die ferne Zukunft ist der Einsatz von *decision trees* anstelle manuell festgelegter Gewichte, die schwer einstellbar und handhabbar sind.

Neben der Erweiterung der Generierung um eine morphologische Komponente, kann auch noch für die Wortstellung die Umgebung in Beispielsätzen ausgeklügelter berücksichtigt werden. Zu prüfen ist, ob die Wortstellung mit Hilfe anderer zielsprachlicher Beispielsätze verbessert werden kann oder ob es stattdessen erfolgsversprechender ist, den Generierungsautomaten hochzurüsten.

Kapitel 7

Implementation

7.1 Allgemeines

Das Programm wurde in C++ entwickelt unter Verwendung der Standard Template Library (STL). Zur Entwicklung wurden folgende Programme eingesetzt:

- Inprise C++ Builder 1.0 (Windows NT)
- GNU C++ Compiler (2.7.2.1,2.8.2) (Linux bzw. Solaris)
- DDD/gdb als Debugger (Linux bzw. Solaris)
- Tcl/Tk 8.0 mit Tix 4.1 für die graphische Oberfläche (NT, Linux, Solaris)

Das Programm selbst läuft unter Linux, Solaris und Windows NT. Allerdings ist unter Windows NT kein POS-Tagger verfügbar, deshalb existiert ein Tagger-Emulator, der den Wörtern die wahrscheinlichste POS zuweist bzw. dem Benutzer erlaubt, die POS abzuändern. Zur Entwicklung wurde überwiegend der C++-Builder von Inprise eingesetzt, da er gegenüber den GNU-Werkzeugen höheren Komfort bietet und bei der Verwendung der STL um ein vielfaches schneller ist.

7.2 Hierarchischer Automat

7.2.1 Aufbau

Die einzelnen Elemente der hierarchischen Automaten sind in Objekten gekapselt. So existieren Klassen für die hierarchische Struktur, die einzelnen endlichen Automaten, für die Knoten in den Automaten sowie für die Übergänge zwischen den Zuständen, etc. Eine Übersicht der Klassen findet sich in Abb. 7.1. Dort sind auch die wichigsten Slots aufgeführt. Die Klasse `TcAutomatonHierarchy` ist die nach außenhin sichtbare Klasse, d.h. die anderen Teile des Programms agieren nur mit Objekten dieser Klasse. Die hierfür gedachten Methoden sind nachfolgend

aufgelistet. Objekte der Klasse `TcAutomatonSet` stellen die einzelnen Verarbeitungsebenen dar, d.h. sie enthalten alle Automaten, die eine hierarchische Ebene bilden. Die Verarbeitung des Eingabestroms in einem Automaten findet innerhalb von Objekten der Klasse `TcAutomatonInstance` statt. Diese Objekte haben Slots für die Speicherung des aktuellen Zustands, wie z.B. der aktuell verarbeitete Übergang und aktuelle Knoten.

Die Bedingungen und Aktionen sind in Objekten gekapselt, die einer Unterklasse der abstrakten Klasse `TcCondition` und `TcAction` angehören. Wie neue Bedingungen und Aktionen hinzugefügt werden können, ist in 7.2.3 beschrieben.

Das Programm ist vom Prinzip her in zwei Teile aufgesplittet. Zuerst wird die Automatendefinition eingelesen und der Automat aus einzelnen Objekten nachgebaut. Anschließend wird der Eingabestrom verarbeitet. Die wichtigsten Methoden, die aufgerufen werden, sind in Abb. 7.2 dargestellt.

7.2.2 Extern nutzbare Methoden

Folgende Methoden der Klasse `TcAutomatonHierarchy` sind für die Verwendung durch andere Komponenten bestimmt:

```
TcAutomatonHierarchy(char* filename)
```

Dieser Konstruktor erzeugt einen neuen hierarchischen Automaten und liest dabei die Automaten-Definition aus der Datei `filename` ein. Das Format, das diese Datei haben muß, ist in Anhang A angegeben. Dieses Format wird auch von dem graphischen Tool, mit dem die Automaten gezeichnet werden können verwendet.

```
void process(TcTokens& input)
bool process(istream& file)
void process(char* c)
```

Diese Methoden verarbeiten mit den Automaten den Eingabestrom. Der Eingabestrom kann dabei entweder aus einzelnen Token, aus einem String oder einem C++-Eingabestrom bestehen. In dem String müssen die Token durch Leerzeichen voneinander getrennt sein, bei dem Eingabestrom in einzelnen Zeilen. Falls der Eingabestrom am Ende ist, gibt `process(istream)` `false`, sonst `true` zurück.

```
TcTokens* getOutputTokens(void)
void print(ostream& out)
char* print(void)
```

Diese Methoden liefern das Resultat eines `process`-Aufrufs zurück, entweder direkt als Token, in einen Ausgabestrom geschrieben oder als String.

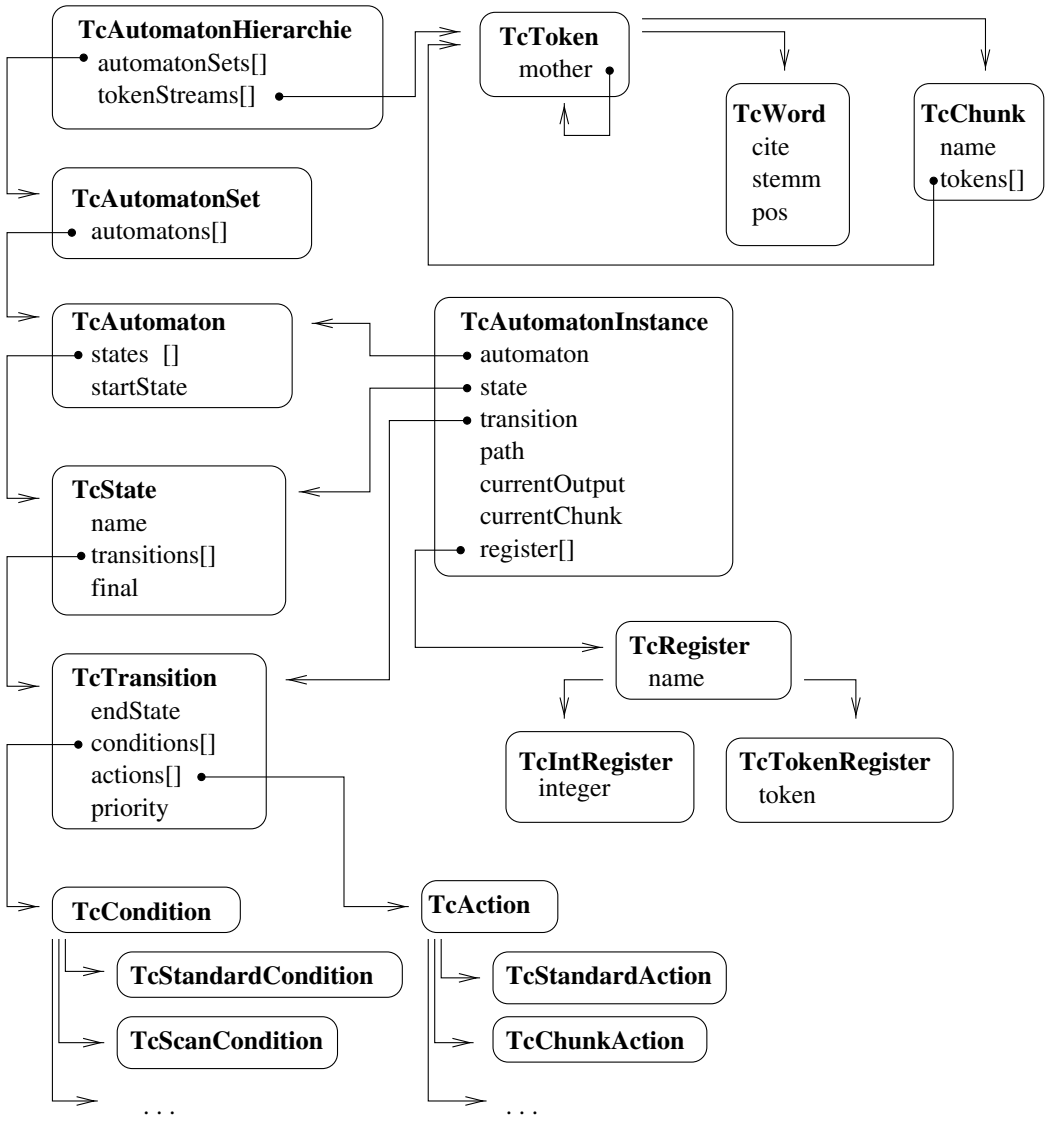

Abbildung 7.1: Die Klassenhierarchie des Automatenmoduls mit den wichtigsten Slots der einzelnen Klassen. Klammern hinter Slotnamen kennzeichen den Slot als Array. Pfeile von Slots zu Klassen kennzeichnen Typzugehörigkeiten, Pfeile von Klassen zu Klassen kennzeichnen die Klassenhierarchie.

Von der Shell aus kann ein hierarchischer Automat so gestartet werden:

`translator p <ha-file> <input-file> <output-file>`

wobei in <ha-file> die Definition des hierarchischen Automaten steht, in <input-file> die Eingabe im *tagged-Format* und das Ergebnis in der Datei <output-file> im *chunked-Format* landet. Die Dateiformate sind in Anhang A beschrieben.

7.2.3 Hinzufügen neuer Bedingungen und Aktionen

Für die Erweiterung der Automaten um neue Bedingungen sind folgende Schritte notwendig:

- Eine Unterklasse von TcCondition muß definiert werden.
- Diese Unterklasse muß folgende Methoden zur Verfügung stellen:

  `static TcCondition* create(char* f,charVec& args [,TcAction** action])`
  Liefert entweder ein neues Objekt der Unterklasse zurück oder NULL, falls der String f nicht der Funktionsname ist. Optional kann noch eine der Bedingung zugehörige Standardaktion in `action` zurückgeliefert werden. Vom Benutzer können dann keine anderen Aktionen mehr angegeben werden, sondern es wird immer diese Aktion ausgeführt.

  `int defaultPriority(void)`
  Gibt die Priorität der Bedingung an, kann zwischen 1 und 1000 liegen. (s.a. Tab. 4.3)

  `bool accept(TcTokensIter& ti,TcAutomatonInstance* ai)`
  Liefert `true` zurück, falls auf dem Eingabestrom `ti` passende Token stehen.

- In der Methode `TcCondition::check` muß noch eine entsprechender Aufruf von `create` eingetragen werden.

Das Hinzufügen von neuen Aktionen läuft analog ab:

- Eine Unterklasse von TcAction muß definiert werden.
- Diese Unterklasse muß folgende Methoden zur Verfügung stellen:

  `static TcCondition* create(char* f,charVec& args)`
  Liefert entweder ein neues Objekt der Unterklasse zurück oder NULL, falls der String f nicht der Aktionsname ist.

  `void process(TcAutomatonInstance* a,TcTokensIter& ti,int tokens)`
  Führt die Aktion aus. Der Eingabestrom ist in `ti` gegeben, die Anzahl der vom Eingabestrom gelesenen Token ist in `tokens` angegeben.

- In der Methode `TcAction::check` muß noch eine entsprechender Aufruf von `create` eingetragen werden.

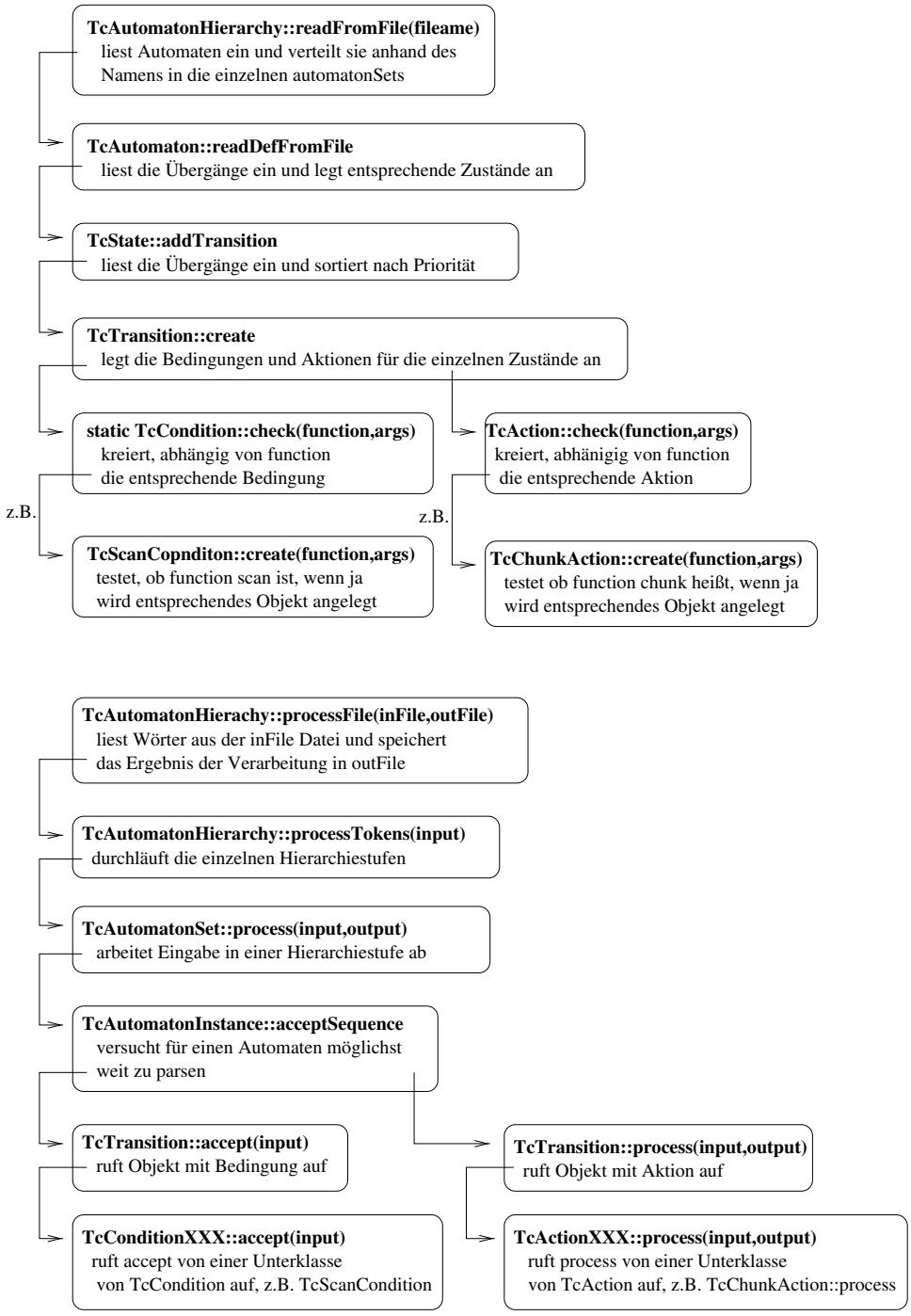

Abbildung 7.2: Die wichtigsten Methoden des Automatenprogramms

7.2.4 Graphische Oberfläche zur Erstellung von Automaten

Zur graphischen Erstellung von hierarchischen Automaten dient ein in Tcl/Tk geschriebenes Tool (siehe Abb. 7.3). Links wird die Automaten-Hierarchie dargestellt. Durch ein Pop-Up-Menü können Ordner und Automaten hinzugefügt, gelöscht, verschoben oder kopiert werden. Rechts befindet sich die Zeichenfläche, in der Automaten editiert werden können. Dieser Programmteil wurde aus [Edw97] weitgehend übernommen, wobei das Programm etwas erweitert und einige Bugs beseitigt wurden. Je nach Anwahl der Buttons links oben können Knoten bzw. Übergänge verschoben, neue Knoten bzw. Übergänge angelegt werden und frei stehender Text, z.B. für Kommentare, eingegeben werden. Mittels eines Pop-Up-Menüs können noch Knoten bzw. Übergänge gelöscht und Start- und Endknoten festgelegt werden.

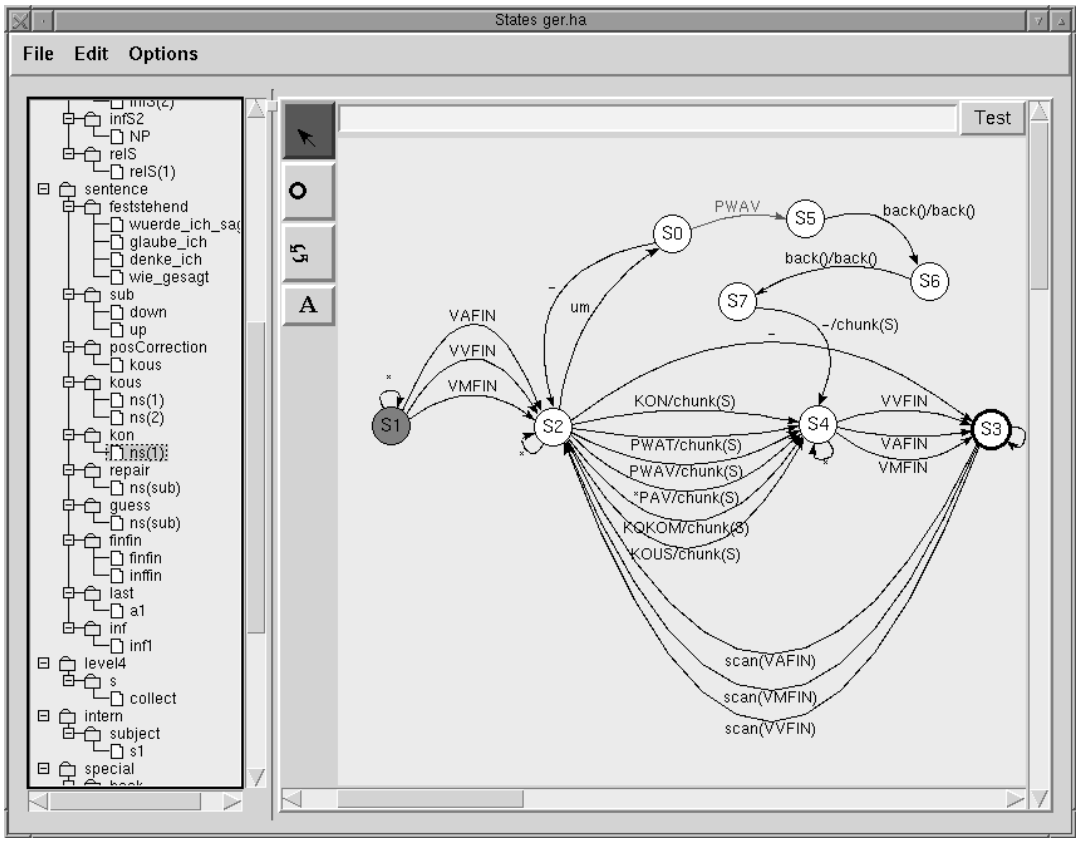

Abbildung 7.3: Das graphische Tool zur Erstellung von hierarchischen Automaten

Zudem existiert noch ein Eingabefeld zum direkten Testen des Automatens, wobei die Ausgabe in einem separaten Fenster ausgegeben wird. Ebenfalls in einem Fenster angezeigt werden die Fehlermeldungen, die das Automatenprogramm ausgibt, z.B. wenn eine nichtdefinierte Funktion aufgerufen wurde oder wenn ein

Syntaxfehler vorliegt. In einem solchen Fall wird zusätzlich der betroffene Automat eingeblendet und der fehlerhafte Übergang rot markiert. Zusätzlich ist in dem Automaten noch eine Endlosschleifendetektion eingebaut. Diese spricht an, sobald ein Automat erfolgreich ist, aber keine Eingabe verarbeitet hat oder wenn ein Zustand innerhalb eines Automatens mehr als 1000 mal besucht wurde. In diesem Fall wird ebenfalls der betroffene Automat angezeigt.

7.3 Template-Generierungs-Modul

7.3.1 Aufbau

Die in Kapitel 4 beschriebenen Funktionen wurden im wesentlichen als Methoden der Klasse `TcAlign` implementiert. Erwähnenswert ist noch die Verwaltung der Links.

1. Die Klasse `TcToken`, die auch im Automat verwendet wird, enthält Slots zur Verwaltung der Links.

2. Ein Objekt der Klasse `TcAlignLink` stellt einen Link dar, wobei die dazugehörigen Token intern verwaltet werden.

Je nachdem was in den jeweiligen Methoden vorteilhafter ist, wird zwischen diesen beiden Konzepten hin- und hergewechselt.

Für einen Aufruf des Alignment-Tools aus anderen Modulen heraus sind folgende Methoden gedacht:

```
TcAlign(char* dictionaryFile)
```

Dieser Konstruktor legt ein Objekt vom Typ `TcAlign` an. Als Wörterbuch beim Alignmentprozeß kommt dabei das in `dictionaryFile` gespeicherte zum Einsatz. Dieses muß sich im *Aligned-Pair-Format* befinden (siehe Anhang A).

```
void alignFile(char* sourceFilename,char* destFilename,
char* indexFilename)
```

Diese Methode führt den eigentlichen Alignierungsprozeß durch. Dabei wird eine Datei `sourceFilename`, die sich im *Chunk-Pair-Format* befinden muß (siehe Anhang A), bearbeitet und das Ergebnis in der Datei `destFilename` im *Aligned-Pair-Format* gespeichert. Außerdem wird eine Indexdatei `indexFilename` im *fPair-Format* angelegt, die ebenfalls von der Transferkomponente verwendet wird.

```
void align(TcChunk* ger,TcChunk* eng)
```

Diese Methode führt das Alignment zwischen den beiden Sätzen durch, die in `ger` und `eng` übergeben werden. Dabei werden die oben erwähnten Slots (wie `linkId`) in den Wörtern und Knoten entsprechend gesetzt.

Von der Shell kann das Alignment mit folgendem Kommando aufgerufen werden:

```
translator a <corpus>
```

Als Eingabe wird dabei die Datei <corpus>.chunked.pair im *Chunked-Pair-Format* genommen, generiert werden die Dateien <corpus>.aligned.pair im *Aligned-Pair-Format* und die Indexdatei <corpus>.fpair. Die Dateiformate werden in Anhang A erläutert.

7.3.2 Graphische Oberfläche des Template-Generierungs-Moduls

Mit Hilfe eines graphischen Tools können die Links, die zwischen den Wörtern angelegt wurden, begutachtet werden (siehe Abb. 7.4). Im linken Fenster kann dabei ein Beispielsatz ausgewählt werden. Zudem können mittels Pop-Up-Menüs noch einzelne Beispielsatzpaare deaktiviert werden, d.h. diese werden zur Übersetzung nicht verwendet. Im unteren rechten Fenster sind beide Sätze mit den Links dargestellt. Die Buchstaben hinter dem Unterstrich geben dabei an, durch welche Methode dieser Link angelegt wurde (siehe dazu auch S.54ff). Im oberen rechten Fenster sind noch detaillierte Informationen über die beiden Sätze dargestellt, die im unteren Fenster keinen Platz mehr haben. Dazu gehören die POS und die Stammform der Wörter und bei bestimmten Chunks die Namen der Automaten, die sie generiert haben.

7.4 Transfer Modul

7.4.1 Aufbau

Die Umsetzung entspricht im wesentlichen der Beschreibung in Kapiel 6. Die dort beschriebenen Funktionen sind dabei als Methoden der Klasse `TcTranslator` realisiert. Eine weitere wichtige Klasse ist `TcTranslationDict`. Deren Objekte enthalten die einzelnen Beispielsatzpaare. Die Beispielsatzpaare selbst werden in Objekten der Klasse `TcChunkPair` gespeichert. Aus Speicherplatzgründen werden die Beispielsatzpaare aber nur bei Bedarf geladen. Zuvor wird lediglich ein Verweis auf den Beispielsatz angelegt, der in einem Objekt der Klasse `TcChunkPairPointer` gespeichert wird. Die externen Wörterbücher sind in Objekten der Klasse `TcExternDict` gespeichert.

Für die Bewertung der Beispielsätze sind eine ganze Reihe von Klassen zuständig: Die Bewertung selbst ist in Objekten der Klasse `TcAssessment` gespeichert.

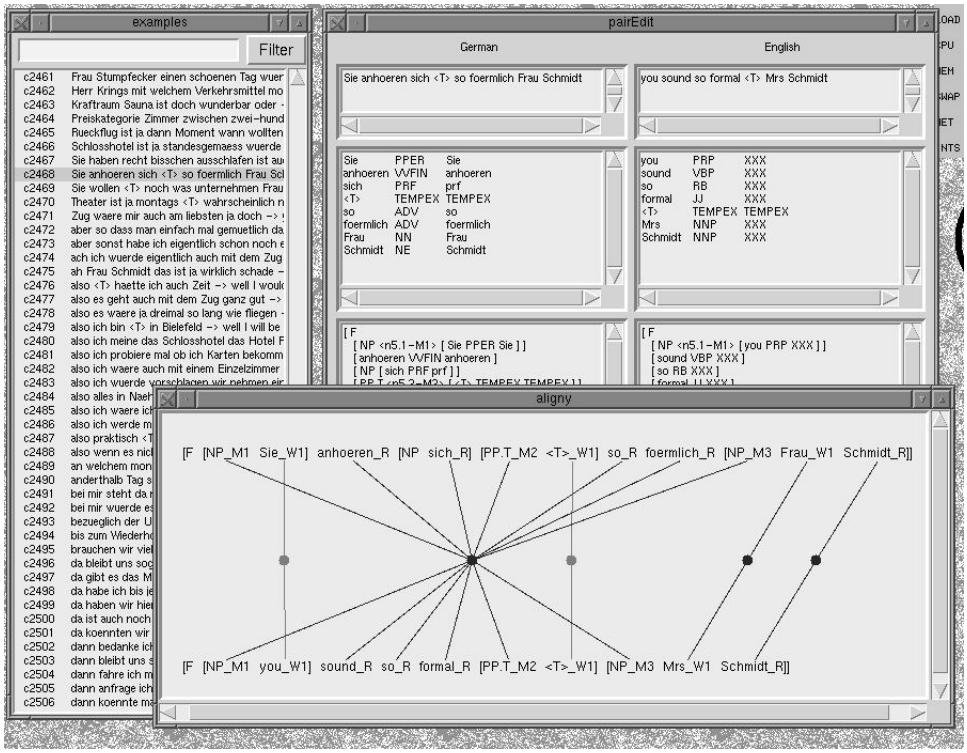

Abbildung 7.4: Das graphische Tool zur Begutachtung des Alignmentprozesses

Während der Bewertung existiert für jedes Token im ausgangssprachlichen Teil des Beispielsatzpaars genau ein solches Bewertungsobjekt. Die auf Seite 69ff beschriebene Tabelle mit Gewichten ist in einem Objekt der Klasse `TcAVTable` abgelegt. Die einzelnen Bedingungen sind in Objekten abgelegt, die eine abstrakte Oberklasse `TcAvCell` haben und werden beim Einlesen der Tabelle erzeugt. Die Elemente mit der die einzelnen Zeilen der Tabelle verglichen werden, sind Objekte, die die abstrakte Oberklasse `TcAVValue` haben. Die möglichen Unterklassen sind in Abb. 7.5 alle aufgeführt.

Eine graphische Übersicht der Klassen mit den wichtigsten Slots ist in Abb. 7.5 dargestellt.

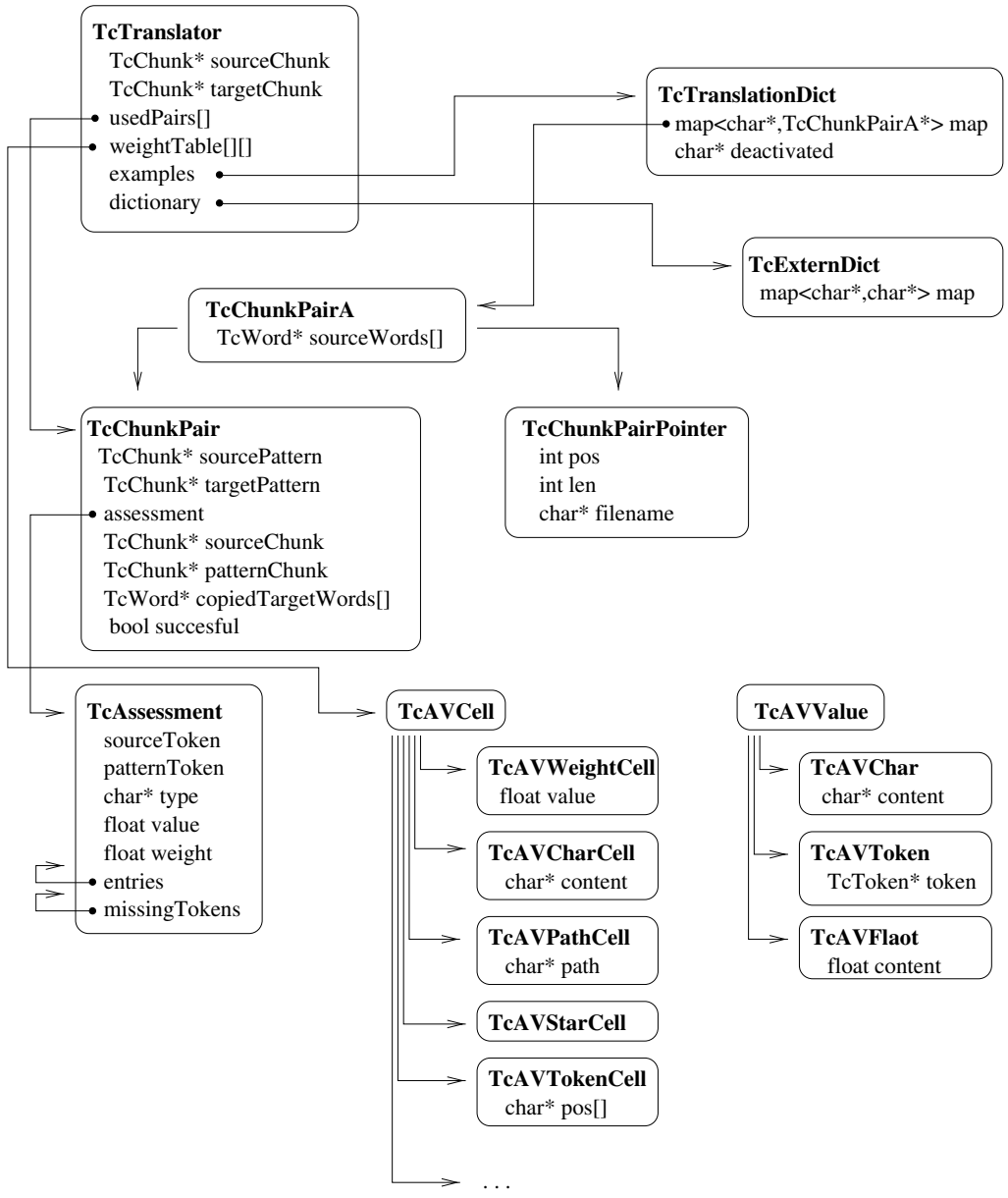

Abbildung 7.5: Die wichtigsten Klassen innerhalb des Transfermoduls

Die Methoden aus der Klasse `TcTranslator`, die andere Module aufrufen können, sind nachfolgend aufgelistet:

`TcTranslator()`

Dieser Konstruktor erzeugt eine neues Transfermodul.

`void useDictionary(char* dictionaryFile)`

Verwendet das Wörterbuch, das in der Datei `dictionaryFile` abgelegt ist. Jedes Wortpaar steht dabei in einer Zeile, als Trenner zwischen ziel- und ausgangssprachlichen Wort dient `--`.

`void useExampleBase(char* examplesFile)`

Verwendet die Beispielsatzpaare aus der Datei `examplesFile`.

`void loadAssessmentList(char* filename)`

Die Datei `filename` enthält die einzelnen Bedingungen zusammen mit den Gewichten. Die Bedingungen innerhalb einer Zeile sind dabei durch Tabulatoren zu trennen.

`void loadDeactivatedList(char* filename)`

Die Datei `filename` enthält eine Liste mit den Bezeichnungen der deaktivierten Beispielsatzpaare. Pro Zeile steht dabei ein Eintrag.

`TcChunk* translate(TcChunk* ger)`

Diese Methode transformiert mit Hilfe der Beispielsatzpaare und des Wörterbuchs die ausgangssprachliche Struktur `ger` in eine zielsprachliche, die als Ergebnis zurückgeliefert wird.

Eine komplette Übersetzung (also einschließlich Generierungsmodul) kann von der Shell mit dem Aufruf erfolgen:

`translate t <corpus>`

Die zu übersetzenden Sätze müssen dabei auf *stdin* geschrieben werden. Von *stdout* kann das Ergebnis gelesen werden. In <corpus> kann der Dateiname des verwendeten Beispielsatzkorpus angegeben werden, der sich im *Aligned-Pair-Format* befinden muß (s. Anhang A).

Falls auch die graphische Oberfläche mit gestartet werden soll, ist der Aufruf

`translate ti <corpus>` angebracht.

7.4.2 Graphische Oberfläche des Transfermoduls

Die Oberfläche dient dazu, das Zustandekommen eines Übersetzungsergebnisses zu visualisieren. Abb. 7.6 zeigt einen Bildschirmabzug. In der obersten Zeile kann der zu übersetzende Satz eingegeben werden, in der Zeile darunter erscheint das Ergebnis. Mit dem *Translate*-Button kann der Übersetzungsprozeß aktiviert werden. Der *Examples*-Button dient dazu, bereits in einer Datei zusammengestellte Sätze übersetzen zu können.

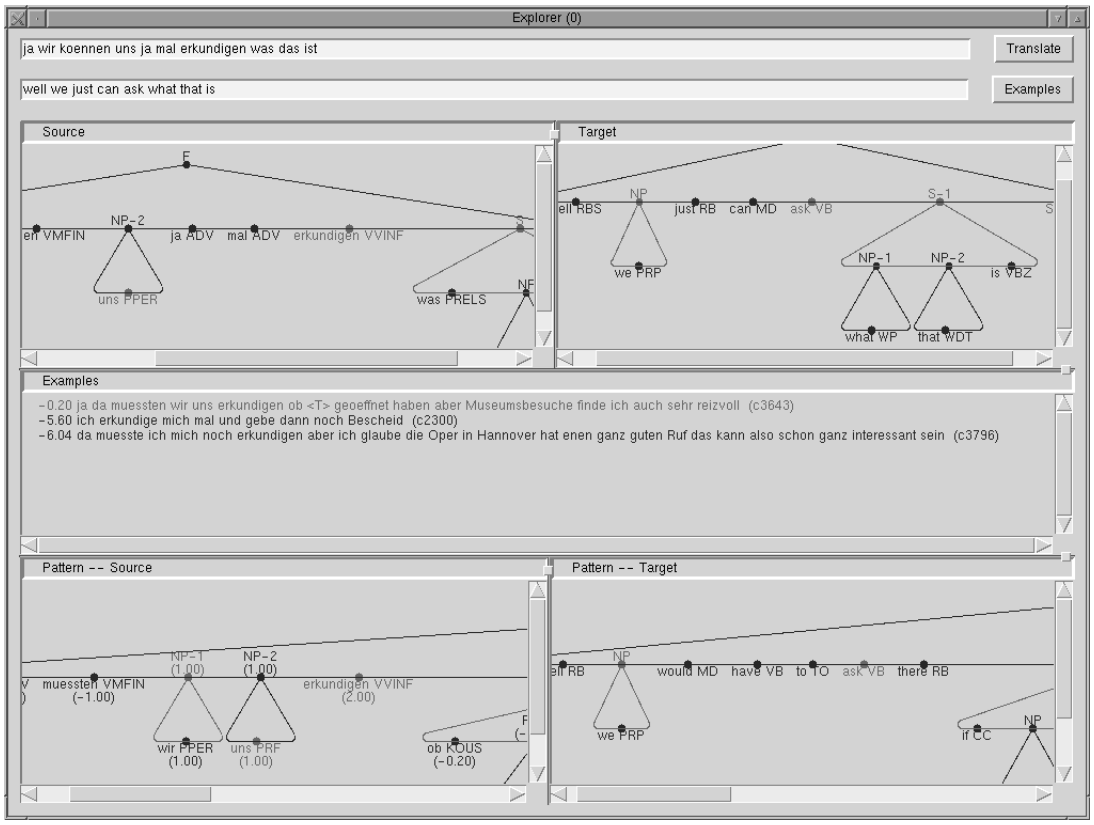

Abbildung 7.6: Die Oberfläche für das Übersetzungsmodul

Die beiden oberen größeren Fenster enthalten den zu übersetzenden Satz und das Ergebnis. In dem mittleren Fenster werden die Bewertungen der einzelnen Beispielsatzpaare angezeigt. Dazu muß in einem der beiden oberen Fenster ein Wort angeklickt werden. Alle Wörter, die zusammen mit diesem Wort übersetzt wurden, werden farblich markiert, ebenso die entsprechenden Übersetzungen. Besonders hervorgehoben (gelb) wird das ausgangssprachliche Wort, das die Auswahl initiiert hat. Wenn ein Beispielsatzpaar angeklickt wird, erscheint im linken unteren Fenster der ausgangssprachliche Satz mit den einzelnen Bewertungen. Blau hervorgehobene Wörter sind Wörter aus dem zu übersetzenden Satz, für die keine Entsprechungen gefunden wurden. Im rechten Fenster ist der entsprechende

zielsprachliche Teil des Beispielsatzpaares zu sehen. Die übersetzten Wörter sind dabei farblich hervorgehoben.

Im mittleren Fenster steht zusätzlich ein Pop-Up-Menü zu Verfügung. Mit dem Menüpunkt *show alignment* stellt das graphische Alignmenttool, falls es gestartet wurde, das Alignment für das ausgewählte Beispielsatzpaar dar. Der Menüpunkt *(de)activate* ermöglicht die Deaktivierung des ausgewählten Beispielsatzpaares, d.h. bei künftigen Übersetzungen wird dieses Beispielsatzpaar nicht mehr berücksichtigt. Der Menüpunkt *edit weights* öffnet ein Fenster, in dem die Gewichte zur Bewertung geändert werden können.

7.5 Generierungsmodul

Das Generierungsmodul ist wie beschrieben implementiert. Die Klasse `TcGen` enthält die dargestellten Methoden.

Methoden, die im Generierungsmodul von anderen Modulen aufgerufen werden können:

`TcGen(char* auto-file)`

Konstruktor für ein neues Generierungsmodul. Der Automat, der bei der Generierung eingesetzt wird, wird aus der Datei `auto-file` geladen.

`generate(TcChunk* ec)`

Diese Methode stellt innerhalb des Knotens `ec` die Wortreihenfolge um.

Dieses Modul hat keine so aufwendige graphische Oberfläche. Allerdings kann im oberen rechten Fenster des Transferfensters ein Pop-Up-Menü aktiviert werden, das es erlaubt, den zielsprachlichen Syntaxbaum vor der Sortierung anzuzeigen bzw. ein Trace-Fenster zu öffnen, das genaue Informationen für jedes Wort bereithält, auf welche Weise es eingefügt wurde.

Kapitel 8

Evaluation

8.1 Korpusbeschreibung

Der verwendete deutsch-englische Korpus wurde im Rahmen des VERBMOBIL-Projekts erstellt und zeichnet sich durch folgende Eigenschaften aus:

- 3892 Satzpaare
- durchschnittliche Anzahl der Wörter pro Satz: im Deutschen 8.3, im Englischen 9.5

Der Korpus liegt in der Form von Dialogen vor. In den Dialogen folgt einer kompletten Äußerung, d.h. bis ein Sprecherwechsel stattfindet, die jeweilige Übersetzung. Äußerungen können somit aus mehreren Sätzen bestehen. Deshalb muß innerhalb der Äußerungen und deren Übersetzungen noch ein Satzalignment vorgenommen werden. Dies geschieht anhand der vorhanden Zeichensetzung. Falls eine Äußerung und deren Übersetzung eine unterschiedliche Anzahl von Sätzen enthalten, wird die komplette Äußerung ignoriert. Andernfalls werden die Sätze der Reihenfolge nach zugeordnet.

Der Schwerpunkt der Evaluation liegt auf den Ursachen der einzelnen Fehler, um Anhaltspunkte zu haben, in welche Richtung das System weiterentwickelt werden kann.

Zunächst wird das Alignment innerhalb der Satzpaare evaluiert und erst anschließend die eigentliche Übersetzung.

8.2 Alignment

Für die Evaluation des Alignments wurden 200 Sätze zufällig ausgewählt. Diese sind in drei Kategorien eingeteilt, da die Ergebnisse je nach Länge der Sätze sehr unterschiedlich sind.

Korpus 1 50 Satzpaare, deren deutscher Teil zwischen 3 und 5 Wörter umfaßt (insgesamt 1750 Satzpaare)

Korpus 2 100 Satzpaare, die eine Länge zwischen 6 und 10 Wörter haben (insgesamt 1000)

Korpus 3 50 Satzpaare, die länger als 10 Wörter sind (insgesamt 892)

Da Satzpaare mit mittlerer Länge die interessantesten Beispielsatzpaare darstellen, wurden doppelt so viele ausgewählt.

Bei der Evaluation des Alignments wurde so vorgegangen, daß für die einzelnen Funktionen, die in Kap. 5 vorgestellt werden, jeweils folgende Fragen untersucht wurden:

- Wie oft kamen die einzelnen Funktionen in den ausgewählten Beispielsatzpaaren zur Anwendung?
- Wie viele Fehler sind dabei aufgetreten und was waren im einzelnen die Ursachen dafür?
- Wie oft wurden die Funktionen nicht aktiviert, obwohl sie eigentlich hätten ansprechen müssen? Dabei sind ebenfalls die Ursachen angegeben.

Die jeweiligen Häufigkeiten zusammen mit den Fehlerquellen sind in Tab. 8.1 aufgelistet. Zunächst werden die einzelnen Fehlerquellen erläutert und anschließend findet die Bewertung der einzelnen Funktionen statt.

Fehlerquellen

In diesem Abschnitt werden die funktionsunabhängigen Fehlerquellen, d.h. Fehlerarten von denen mehrere Funktionen betroffen sind, vorgestellt. Fehlerquellen, die nur eine Funktion betreffen, sind im nächsten Abschnitt bei der entsprechenden Funktion erläutert. In Klammern sind noch die Abkürzungen angegeben, die in Tab. 8.1 verwendet werden.

Korpus (K)

In den untersuchten Sätzen war lediglich in einem Fall eine zu freie Übersetzung Ursache falscher Kopplungen und zweimal war das Satzalignment nicht korrekt, d.h. die Satzgrenzen waren nicht identisch zwischen Original und Übersetzung.

Part-of-Speech-Tagger (POS)

Der deutsche POS-Tagger hat u.a. Schwierigkeiten bei der Unterscheidung zwischen den verschiedenen Verbformen (finit, infinit, etc.), was bei der Bestimmung der Nebensatzgrenzen Folgefehler verursachen kann. Weitere Problembereiche

Funk.	ange.	erfolg.	nicht erfolgreich		nicht erkannt	
			insg.	Ursachen	insg.	Ursachen
Korpus 1 (50 Sätze)						
D	114	112	2	fÜ:1 P:1	1	fE:1
CP	8	7	1	POS:1	2	POS:1 dd:1
WP	5	5	0		0	
SL	4	3	1	Tel:1	2	dd:1 dd+Tel:1
R	50	45	5	K:1 Tel:2 W:2		
Korpus 2 (100 Sätze)						
D	373	367	6	fZ:2 fÜ:2 fE:2	8	fE:7 POS:1
CP	32	25	7	Tel:1 ST:1 POS:2 NP:1 PP:1 P:1	2	dd:2
WP	17	11	6	W:3 POS:2 P:1	3	ST:2 dd:1
SL	10	7	3	Tel:2 NP:1	13	Tel:2 ST:1 NP:2 PP:1 dd:5 P:2
R	89	71	18	Tel:8 NP:1 PP:1 W:4 WP:2 CP:1 P:1		
Korpus 3 (50 Sätze)						
D	330	326	4	P:2 fÜ:1 SG:1	6	fE:6
CP	24	20	4	POS:2 P:2	2	dd:1 NP:1
WP	8	6	2	Tel:1 P:1	0	
SL	9	7	2	NP:1 Tel:1	6	K:1 POS:1 Tel:1 PP:1 dd:2
R	30	17	13	Tel:9 D:3 fÜ:1		

Tabelle 8.1: Aufstellung der Häufigkeiten, wie oft eine Funktion angewendet wurde, wieviele Fehler dabei aufgetreten sind und wie oft die Funktion nicht aktiviert wurde. Die Funktionsabkürzungen bedeuten im einzelnen: `D=alignWithDict`, `CP=alignSubChunkPerPos`, `WP=alignWordsPerPos`, `SL=alignChunkSameLevel`, `R=alignRest`

sind noch die Unterscheidung zwischen Artikel (ART) und Demonstrativpronomen (PDS) und zwischen Adverb (ADV) und adverbial gebrauchten Adjektiv (ADJD). Beides sind Ursachen für inkorrekte NP-Bestimmungen.

Beim englischen POS-Tagger treten häufiger Fehler auf bei der Unterscheidung zwischen Nomen und Verb (z.B. *at which date(VBP)*), zwischen Verb und Konjunktion (z.B. *like*) und zwischen Konjunktionen und Demonstrativpronomen (z.B. *that*).

Zeitausdrücke (Tel)

Zeitausdrücke werden von dem verwendeten Tool nicht vollständig erkannt, z.B.

in tempex([from:dom:1]) Oktoberhälfte →
tempex([from:dom:1]) half of tempex([from:month:oct])

oder im Deutschen und Englischen unterschiedlich behandelt, z.B.

Abendprogram → program for tempex([from:pod:evening])

Eine weitere Schwierigkeit stellen Zahlen dar, die meist als Zeitausdrücke erkannt werden und die korrekte Erkennung von Nominalphrasen stören, ein Beispiel hierfür:

ich buche da zwei Einzelzimmer wird zu
ich buche tempex([from:tod:2:0]) Einzelzimmer

Nebensatzgrenzen (ST)

Die Nebensatzerkennung in der syntaktischen Analyse hat nicht richtig funktioniert, infolgedessen kam es zu Zuordnungsfehlern. Vorallem die aufgestellte englische Grammatik hat sehr große Probleme, da eine Konjunktion zwischen Haupt- und Nebensatz stehen muß. Nebensatzgrenzen ohne Konjunktionen zu finden, ist im Englischen sehr schwierig und hätte den Rahmen dieser Arbeit überschritten. Die Erfolgsquote der Satztrennung ist in Tab. 8.2 aufgelistet. Wenn in beiden Sätzen eine unterschiedliche Anzahl von Nebensätzen erkannt wurde, werden diese Sätze vorsichtshalber aussortiert. In Tab. 8.2 sind diese Sätze in der Spalte *Fehlkopplungen* nicht mehr enthalten.

Nominalphrase (NP)

Eine Nominalphrase wurde nicht korrekt erkannt. Z.B. können längere Eigennamen häufiger nicht als eine NP erkannt werden:

[NP Reiseagentur(NN) Maritim(NE)] [NP Fischenbeck(NN)]

In diesem Fall kommt erschwerend hinzu, daß die NP anders als im Englischen aufgesplittet wird:

Korpus	Größe	mit Nebensätzen	erfolgreich	nicht erfolg. (d.+e.,d.,e.)	Fehlkopplungen	Korpusfehler
1	50	0	0	0	0	0
2	100	23	12	11 (2,4,5)	1	0
3	50	31	8	21 (9,1,11)	10	2

Tabelle 8.2: Die Erfolgsquote der Satztrennungsautomaten

[NP travel(NN) agency(NN)] [NP Maritim(NNP) Fischenbeck(NNP)]

Eine weitere Fehlerquelle besteht darin, daß grundsätzlich kein Attachment von PPs an NPs stattfindet. Besonders problematisch ist dies, wenn in einer Sprache die PP gar nicht vorhanden ist, ein Beispiel hierfür ist:

[NP die Fahrtdauer] → *[NP the duration] [PP of the trip]*

Präpositionalphrase (PP)

Eine Präpositionalphrase wurde nicht korrekt erkannt, z.B.

as far [PP as transportation]

In diesem Fall fehlt eine entsprechende Behandlung in der Grammatik.

Infinitivsatz (IS)

Infinitivsätze werden im Rahmen dieser Arbeit nicht behandelt und verursachten dadurch Fehler beim Alignment.

doppeldeutig (dd)

Ein Wort bzw. eine Phrase konnte nicht eindeutig zugeordnet werden, da mehrere alternative Kopplungen möglich sind. Ein Beispiel hierfür ist

bei mir steht da nämlich auch schon [NP eine Dienstreise] drin
[NP there] is also already [NP a business trip] in there for me then

wobei *eine Dienstreise* weder *there* noch *a buisness trip* zugeordnet werden kann, da die Funktion `alignChunkPerPOS` eine gleiche Anzahl von Wörtern erwartet und die Funktion `alignChunkSameLevel` zwei nicht gekoppelte NPs zur Verfügung stehen.

Bewertung der Funktionen

In diesem Abschnitt werden die Ergebnisse für die einzelnen Funktionen aus Tab. 8.1 kommentiert. Außerdem werden noch funktionsspezifische Fehlerquellen, die im letzten Abschnitt nicht erwähnt wurden, vorgestellt.

AlignWithDict (D)

Die Zuordnung kann aus mehreren Gründen fehlschlagen bzw. gar nicht stattfinden.

Falsche Zuordnung (fZ)
Da `alignWithDict` die Wörter auch zuordnet, wenn sie nicht eindeutig gekoppelt werden können, wobei in diesem Fall die Reihenfolge entscheidend ist, können dadurch die falschen Wörter gekoppelt werden. Dies kommt glücklicherweise sehr selten vor. Der positive Effekt überwiegt, im 2.Korpus wurden 9 nicht eindeutig gekoppelt und es sind dabei nur 2 Fehler aufgetreten.

fehlende Übersetzung (fÜ)
Für bestimmte deutsche Wörter ist im Wörterbuch angegeben, daß keine englische Übersetzung vorhanden sein muß (z.B. *ja*,*dann*). Falls die Liste mit den möglichen Übersetzungen nicht vollständig ist, kann es vorkommen, daß fälschlicherweise dieses Wort mit keinem englischen Wort gekoppelt wird. Das kann zu Übersetzungsfehlern führen, da das entsprechende englische Wort mit Hilfe anderer Funktionen gelinkt wird.

fehlender Eintrag (fE)
Fehlender Eintrag im Wörterbuch, meist handelt es sich dabei um nicht verzeichnete Ausnahmen wie

nicht wahr → *right* oder *es gibt* → *there are*

Teilweise führt dies zu Fehlern, z.B. wenn ein *da* im Satz fälschlicherweise mit *there* gekoppelt wird. Meistens werden jedoch nur größere Einheiten zusammengelinkt, was keine direkten Übersetzungsfehler zur Folge hat. Es verursacht jedoch eine schlechtere Abdeckung, was wiederum indirekt zu Fehlern führen kann.

Satzgrenze (SG)
Kritisch ist, wenn für ein Wort zufällig in einem anderen Satzteil dessen Übersetzung steht, nicht aber im eigenen wie z.B. in

und da wollte ich mit Ihnen einen Termin ausmachen weil Sie sollen mich dahin begleiten

and I wanted to arrange a date for that with you because you are supposed to accompany me there

der Fall, bei dem *da* und *there* unpassenderweise zusammengelinkt wurden. Dadurch funktioniert die Zuordnung der Nebensatzknoten nicht mehr korrekt, da damit eine Voraussetzung für die Kopplung verletzt ist (siehe auch Beschreibung von `searchChunk`, S. 54). Dies trat allerdings nur einmal in den 200 Testsatzpaaren auf.

Zusammenfassend läßt sich sagen, daß das Wortalignment recht gut funktioniert. Der Verzicht auf die Eindeutigkeits-Beschränkung wirkt sich positiv aus, da dadurch mehr Wörter gelinkt werden konnten.

AlignChunkPerPOS (CP)

Dieser Funktion liegt die Annahme zugrunde, daß ausgangs- und zielsprachliche Knoten mit gleicher Anzahl von Wörtern und entsprechendenen POS Übersetzungen sind. Dies gilt natürlich nur mit Einschränkungen, ein Beispiel hierfür ist:

anhört [NP sich] auch eigentlich ganz gut ne
[NP it] also sounds quite good actually right

wobei *sich* und *it* gekoppelt werden. Ob sich eine solche Kopplung auch als Übersetzungsfehler bemerkbar macht, hängt dann von den restlichen Beispielsätzen ab. Prinzipielle Fehler dieser Art sind in Tab. 8.1 mit *P* markiert.

Die Tests haben gezeigt, daß die Fehler in der syntaktischen Analyse die Funktion weit mehr beeinträchtigen als prinzipbedingte Fehler. Im großen und ganzen arbeitet die Funktion sehr zufriedenstellend, sie kam häufig zum Einsatz und dies bei geringer Fehlerquote.

AlignWordPerPOS (WP)

Diese Funktion basiert auf der Annahme, daß Wörter mit vergleichbarer POS Übersetzungen sein könnten, wenn eine eindeutige Zuordnung möglich ist. Auch dies muß nicht immer zutreffen, z.B. bei dem Satzpaar

bei mir steht da nämlich auch schon eine Dienstreise drin
there is also already a business trip in there for me then

bei dem *nämlich(ADV)* und *there(RB)* gekoppelt werden.

Als Ergebnis läßt sich festhalten, daß die Anzahl der Wörter, die mit Hilfe dieser Funktion gekoppelt werden können, eher bescheiden ist und dies bei einer recht hohen Fehlerquote (ca. 30%). Dies ist stärker durch Fehler des POS-Taggers und Problemen beim Einsatz des Wörterbuchs bedingt als durch prinzipielle Probleme, die ebenfalls in der Tab. 8.1 mit *P* markiert sind.

AlignChunkSameLevel (SL)

Diese Funktion koppelt bislang nicht gekoppelte Knoten mit gleicher Bezeichung, wenn sie eindeutig zugeordnet werden können unter der Annahme, daß diese Übersetzungen sind. Dies muß nicht immer zutreffen, in den untersuchten Testpaaren kam das allerdings nicht vor. Durch die Beschränkung der POS innerhalb der Knoten werden aber zweimal Übersetzungen nicht gelinkt.

[NP ein einigermassen(ADV) interessantes Nachtleben]
[NP a somewhat interesting night life]

Die Funktion arbeitet im großen und ganzen recht brauchbar, wenn auch viele Kopplungen wegen Fehler in der syntaktischen Analyse und fehlender Eindeutigkeit nicht durchgeführt werden können.

AlignRest (R)

Die Funktion koppelt alle bislang noch nicht gelinkten Wörter zusammen. Dadurch machen sich die Fehler von anderen Funktionen hier bemerkbar, was ebenfalls in Tabelle 8.1 aufgeschlüsselt ist (*D,WP,CP*). Außerdem leidet diese Funktion unter der hohen Fehlerhäufigkeit der syntaktischen Analyse, vor allem wenn Zeitausdrücke nicht richtig erkannt werden und noch Teile des nur teilweise erkannten Zeitausdrucks mitgelinkt werden.

Ein besonderer Fall, der ein prinzipielles Problem (mit *P* vermerkt) darstellt, ist:

ja ich würde dann wohl <T> nehmen
well I guess I will take <T> then

Schwierig hierbei ist, daß dem deutschen *ich* zwei englische *I* entsprechen, aber nur eins gekoppelt wird. Dies macht sich als Übersetzungsfehler bemerkbar, sobald ein andere Subjekt als *ich* verwendet wird.

Eine weitere Besonderheit ist, daß Satztrennungsfehler sich negativ auswirken, da sich dann mehrere Verben in einem Knoten befinden, was eine korrekte Zuordnung von Subjekt und Objekten bei dem hier verwendeten Verfahren unmöglich macht. Allerdings läßt sich dies immer dann erkennen, wenn im ausgangs- und zielsprachlichen Satz eine unterschiedliche Anzahl von Nebensätzen erkannt wird. Auf diese Weise werden 21 Sätze aussortiert, 11 Sätze werden fälschlicherweise nicht aussortiert (s.a. Tab. 8.2).

Solange die Zeitausdrücke richtig erkannt werden und die Wörter über das Wörterbuch korrekt gekoppelt werden, funktioniert die Funktion prinzipiell recht gut.

Fazit

Zur Zeit ist eigentlich eine manuelle Nachbearbeitung der Sätze notwendig, die allerdings für den nachfolgenden Test der Übersetzungskomponente nicht durchgeführt wurde. Die Fehlerursache liegt aber stärker bei der syntaktischen Analyse als bei prinzipiellen Problemen der einzelnen Funktionen. Ob dies bei schwierigeren und anspruchsvolleren Sätzen auch so bleibt, kann nur schwer abgeschätzt werden.

Hilfreich für die manuelle Nachbearbeitung wäre auf jeden Fall eine Funktion, die potentielle Kandidaten für fehlerhaftes Alignment aufspürt, so daß die Durchsicht nur für einen Teil der Sätze notwendig ist.

8.3 Übersetzung

Für die Evaluation der Übersetzung wurden die 100 Sätze mittlerer Länge aus Korpus 2, die auch bei der Evaluation des Alignments eingesetzt wurden, mittels leave-one-out bewertet, d.h. der untersuchte Satz wurde beim Übersetzen aus dem Beispielkorpus herausgenommen. Zur Übersetzung wurde auf den kompletten Beispielsatzkorpus mit 3800 Sätzen zurückgegriffen. Korpus 3 mit den längeren Sätzen wurde nicht evaluiert. Problematisch an diesem Korpus ist weniger die Länge der Sätze, sondern deren Inhalt. Die einzelnen Sätze stimmen inhaltlich zu wenig mit dem restlichen Korpus überein und sind somit für das leave-one-out Verfahren nicht sonderlich geeignet. Korpus 1 mit den kurzen Sätzen wurde als nicht interessant genug eingestuft.

Fehlerquellen

Analog zu der Evaluation des Alignments wurden die einzelnen Fehler in Kategorien eingeteilt. Wie häufig einzelne Fehlerarten aufgetreten sind, ist in Tabelle 8.3 vermerkt.

Korpus (K)

Dabei handelt es sich um Fehler, die auf einen unzureichenden bzw. fehlerhaften Korpus zurückzuführen sind. Die Fehlerarten lassen sich dabei in folgende Kategorien einteilen:

nicht vorhandenes Wort (M) Ein Wort bzw. ein Ausdruck kommt weder im Korpus, noch im externen Wörterbuch vor.

unpassender Wörterbucheintrag (W) Ein Wort bzw. ein Ausdruck kommt nicht im Korpus vor, aber in dem externen Wörterbuch. Allerdings ist die dort angegebene Übersetzung in diesem Kontext nicht die passende.

unpassende Beispielsatzpaare (B) Für ein Wort bzw. ein Ausdruck sind zwar Beispielsatzpaare im Korpus vorhanden, aber mit in diesem Fall unpassenden Übersetzungen. Falls ein Beispielsatzpaar mit einer zum Kontext passenden Übersetzung vorhanden ist, aber nicht ausgewählt wurde, fällt der Fehler nicht in diese Kategorie, sondern in die Kategorie Lookup-Fehler (L).

Redewendung nicht vorhanden (R) Eine Redewendung bzw. feststehender Ausdruck ist nicht im Korpus vorhanden und wird deshalb falsch übersetzt.

Fehler in Korpus (F) Eine falsche Übersetzung innerhalb des Beispielsatzkorpus war Ursache für einen Übersetzungsfehler.

Kategorie	insg.	Einzelursachen
Korpus (K)	26	M:10,W:2,B:9,R:2,F:3
Vorverarbeitung (V)	21	Tel:5,ST:1,SG:3,WP:4,CP:5,NP:2,fÜ:1
Tel-Erkennung (T)	13	Z:6 T:7
syntaktische Analyse (S)	19	POS:7,NP:2,ST:6,I:2,fF:2
Lookup (L)	9	G:4,Z:2,H:1,E:2
Generierung (G)	32	U:1, A:11, eF:1, kF:5, M:9 TG:5
morphologische Anpassung (G-M)	9	VZ:2,VF:2,S:4,P:1

Tabelle 8.3: Die verschiedenen Fehlerarten, die bei dem Testkorpus aufgetreten sind, zusammen mit ihren Häufigkeiten (100 Testsätze insgesamt)

Ein Beispiel aus dem Testkorpus für das Fehlen eines passenden Beispielsatzpaars ist die Übersetzung von *hinkommen* in dem Satz

hundert-neun-zehn Mark, das sollte wohl hinkommen

wobei als Beispielsatzpaar unpassenderweise

wie sollen wir da hinkommen → how should we get there

ausgewählt wurde.

Häufig fehlen auch in den Beispielsätzen die entsprechenden Präpositionen, wie z.B. für die Übersetzung von *buchen* in dem Satz

das Hotel kann ich Ihnen buchen

lediglich Beispielsätze ohne indirektes Objekt vorhanden sind und somit inkorrekt zu

I can book the hotel you statt *I can book the hotel for you*

übersetzt wird. Der umgekehrte Fall, daß nur Versionen mit Präposition im Beispielsatzkorpus vorkommen, stellt normalerweise kein Problem dar, da die Präposition miteinander gelinkt sind und somit nicht übersetzt werden. Die Ausnahme, daß aus ausgangssprachlichen Objekten zielsprachliche PPs werden (und somit die Präposition mit dem Verb übersetzt wird), wird insofern von der Generierung behandelt, als daß die Präposition bei nicht vorhandener NP einfach gestrichen wird.

Vorverarbeitung (V)

Bei der Vorverarbeitung, d.h. bei dem Alignment der Beispielsatzpaare, ist bereits ein Fehler aufgetreten. Die detaillierte Aufteilung entspricht den in Abschnitt 8.2 erläuterten Fehlerquellen.

Tel-Erkennung (T,Z)

Die Komponente zur Erkennung von Zeitausdrücken hat entweder eine Zahl innerhalb einer Nominalphrase als Zeitausdruck erkannt (Z) oder der Zeitausdruck wurde nicht in einen TEL-Ausdruck umgewandelt (T).

Syntaktische Analyse (S)

Die syntaktische Analyse hat den zu übersetzenden Satzes nicht korrekt verarbeitet. Die einzelnen Ursachen entsprechen den in Abschnitt 8.2 aufgelisteten, wie z.B. POS-Fehler, inkorrekte NP-Erkennung etc. Besonders gravierend wirkt sich aus, wenn Nebensatzgrenzen nicht erkannt werden, da die NPs und PPs nicht mehr richtig den Verben zugeordnet werden können und die Generierungskomponente nicht mehr in der Lage ist, die Wörter bzw. Knoten korrekt zu ordnen. Ein Beispiel ist der Satz

dann fragt sich jetzt nur noch ob wir abends miteinander ausgehen

der zu

then you only again together go out asks we now if in the evening

übersetzt wird.

Infinitivsätze (I) werden (noch) nicht korrekt behandelt. Mehrere Probleme treten dabei auf und verhindern eine einfache Berücksichtigung. Bereits die Trennung von Infinitivsatz und Hauptsatz ist nicht immer eindeutig möglich, außerdem müßte das Subjekt, das von der syntaktischen Komponente gar nicht ermittelt wird, in den Infintivsatz eingeblendet werden, um eine kontextabhängige Übersetzung der Wörter im Infintivsatz zu ermöglichen.

Weitere in der Grammatik nicht behandelte Phänomene kommen zweimal vor (fF): *eines der ...* und *... noch etwas zu besprechen ist* konnten nicht korrekt gehandhabt werden.

Lookup-Fehler (L)

Die Auswahlfunktion, die den passendsten Beispielsatz sucht, hat den falschen Beispielsatz ausgewählt bzw. bei der Zuordnung der Wörter oder Phrasen fehlerhaft gearbeitet. Folgende Unterkategorien existieren:

Gewichtung (G) Die Gewichtung ist für den getesteten Satz falsch eingestellt.

Zuordnung (Z) Die Einteilung der NPs in Subjekt bzw. Objekt wird nicht von der syntaktischen Analyse, sondern implizit von der Auswahlfunktion vorgenommen und das nicht immer fehlerfrei.

fehlende Hierarchie (H) Die (noch) nicht eingebaute semantische Abstandsfunktion zwischen Wörtern führt zur Auswahl des falschen Beispielsatzes.

fehlende Satzeinteilung Es fehlt eine Möglichkeit für die einzelnen Sätze anzugeben, inwieweit sie abstrahiert werden können.

Bei der Durchsicht aller getesten Beispielsätze konnte keine krasse Fehlgewichtung festgestellt werden, die Probleme beschränken sich eher auf Spezialfälle.

So liegt ein Problem bei der Bewertung darin, daß bei der gleichzeitigen Übersetzung mehrerer Wörter der Mittelwert der einzelnen Bewertungen genommen wird. Dies ist notwendig, da sonst bei einer einfachen Aufaddierung in vielen Fällen diese gegenüber anderen Sätzen zu gut abschneiden würden. Manchmal wirkt sich dies aber auch negativ aus, so z.B. zum Übersetzen des Verbs *heisst* in dem Satz[1]

das heisst Martim

das Satzpaar

das(1.0) heisst(2.0) <T>(-1.0) -NP(-1.5) → that means <T>

dem Satzpaar

das(1.0) heisst(1.0) Martim-Hotel(-1.0) Hannover(-1.0) →
it is called Martim-Hotel

vorgezogen wird. Dies liegt daran, daß im ersten Fall *das heisst* und *that means* zusammengekoppelt sind, aber im zweiten Fall nur *heisst* und *is called*. In dem ersten Satz wird deshalb die negative NP-Bewertung nur halb verrechnet, da sich nur bei der Bewertung des Verbes die fehlende NP negativ bemerkbar macht, aber nicht bei dem Wort *das* (s.a. Tab. 6.1). Außerdem wird bei der Übersetzung von *das* das Verb stärker berücksichtigt, so daß im ersten Satz das Verb mit

[1] Die Werte in Klammern geben die vergebene gewichtete Bewertung wieder.

$(1.0 + 3.0)/2 = 2.0$ und im zweiten mit 1.0 gewertet wird. Dieses Beispiel zeigt deutlich die Schwierigkeit alle Effekte bei der manuellen Vergabe von Gewichten zu berücksichtigen.

Ein weiters Problemfeld ist die Subjekt-/Objekt-Zuordnung. Sie tritt z.B. bei dem Testsatz

und [NP was] sollte [NP man] da noch besprechen

und dem ausgewählten Beispielsatzpaar

... und dann besprechen [NP wir] noch [NP Genaueres] →
... and then [NP we] will discuss [NP further details]

auf, wobei *[NP was] [NP wir]* und *[NP man] [NP Genaueres]* zugeordnet wird. Die Bewertungen für diese Zuordnung und die andere mögliche Zuordnung sind gleich (jeweils -1.0), so daß die Reihenfolge der beiden NPs über die Zuordnung entscheidet. Eine Einteilung der Wörter in semantische Kategorien würde in diesem Fall vermutlich zu einer korrekten Zuordnung führen.

Ein weiteres Beispiel ist die Übersetzung von *steht* in dem Satz

bei mir steht da nämlich auch schon eine Dienstreise drin

mit Hilfe des Beispielsatzpaars

und zwar stehen uns da Hotels zur Verfügung →
and in fact there are hotels available to us

Hier wird aus den gleichen Gründen die NP *eine Dienstreise* der NP *uns* statt der NP *Hotels* zugeordnet, was bei der Generierung eine falsche Wortstellung verursacht, da das Subjekt leer bleibt und *eine Dienstreise* als direktes Objekt behandelt wird. Auch hier könnte eine Worthierarchie den Fehler beheben.

Ein weiteres Beispiel, bei dem die Verwendung von semantischen Kategorien vorteilthaft ist, ist die Übersetzung von *reservieren*, wobei der zu übersetzende Satz

soll ich Ihnen denn schon mal im Hotel ein Zimmer reservieren

lautet und

soll(1.0) ich(1.0) die(-0.3) Karten(-1.0) reservieren(1.0) -PP(-1.5)
-NP(-3.0)

ausgewählt wurde, statt

könnten(-1.0) sie(-1.0) mir(-1.0) da(-0.3) was(-1.0) reservieren(1.0)
→
could you book something for me there

Dieser Satz hätte von der Struktur besser gepaßt, da auch ein indirektes Objekt vorhanden ist. So wurde *Ihnen* fälschlicherweise direkt mit *you* übersetzt, während im zweiten Satzpaar *Ihnen* korrekt mit *for you* übersetzt worden wäre.

Problematischer sind Fälle wie z.B. *wenn*, das als *if* oder *when* übersetzt werden kann und das nicht durch einfache Kontextanalysen entschieden werden kann.

Um mit freieren Übersetzungen und Redewendungen besser umgehen zu können, müßten die Sätze noch eingeteilt werden, wie gut sie sich abstrahieren lassen. Ein Beispiel im Testkorpus ist die Übersetzung von *es gibt* in dem Satz

da gibt es das Maritim-Hotel in der Nähe vom Hauptbahnhof

auf. Statt z.B. das vorkommende Beispielsatzpaar

also(-0.3) was(-1.0) gibt(1.0) [MP es(1.0)] denn(-0.3) da(0.3) →
[NP what] is there then

wird unglücklicherweise das Beispielsatzpaar

ja(-0.3) [NP was(-1.0)] gibt(2.0) [NP es(1.0)] →
well [NP what] do [NP they] have

ausgewählt, das eine etwas freiere Übersetzung beinhaltet und in diesem Kontext zu einer inkorrekten Übersetzung führt. Die Gewichtung für *was* zu ändern, um dieses Problem zu beheben, ist nicht besonders sinnvoll, da, wie im ersten Satz zu sehen ist, *was* im allgemeinen durch eine andere NP ersetzt werden kann.

Ein anderes Beispiel ist die Übersetzung von *sind* in dem Testsatz

... wenn wir dann da sind.

Statt den in diesem Fall passenden

da(0.3) bin(0.9) ich(-1.0) [PP im Urlaub](-1.0) →
I will be on vaction then

wird der kurze Satz, der im Dialog elliptisch verwendet wird

bin(0.9) ich(-1.0) → *I will*

ausgewählt, da *da* und *im Urlaub* nicht zusammengelinkt werden. Somit wird *sind* nur mit *will* übersetzt, statt mit *am*, das von einer (noch nicht vorhandenen) morphologischen Komponente entsprechend angepaßt werden würde.

Generierungs-Fehler (G)

Die Wortreihenfolge ist nicht korrekt, dies kann folgende Ursachen haben:

Einsortierung mittels Umgebung (U) Die Einsortierung unter Berücksichtigung der POS der angrenzenden Wörter führt in diesem Fall zu einem Fehler

Generierungsautomaten (A) In den Generierungsautomaten wird ein Phänomen nicht behandelt bzw. ist ein Fehler vorhanden

Fragesatz fehlerhaft erkannt (eF bzw. kF) Der zu übersetzende Satz wurde nicht korrekt als Fragesatz erkannt (eF) oder er wurde fälschlicherweise als Fragesatz eingestuft (kF).

Morphologie falsch (M) Dadurch, daß die morphologische Komponente fehlt, die Beispielsätze aber auch anhand der Stammform herausgesucht werden können, um eine bessere Abdeckung zu erreichen, können Verben falsch konjungiert sein (VF) bzw. in der falschen Zeit (VZ) stehen und bei Nomen der Numerus inkorrekt sein (S,P).

TEL-Generierung Zeitausdrücke wurden von dem separaten Tool nicht korrekt ins Englische übersetzt

Fazit

Zur Zeit sind noch überall Verbesserungen notwendig, vorallem ein umfangreicherer und besser abdeckender Beispielsatzkorpus scheint angeraten. Aber auch die syntaktische Analyse und Generierung bedarf der Weiterentwicklung. Es ist eben ein weiter Weg bis zu einem brauchbaren Übersetzungssystem ...

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Das Ziel dieser Diplomarbeit, ein beispielbasiertes Übersetzungssystem zu entwickeln, wurde prinzipiell erreicht. Ein praktischer Einsatz im VERBMOBIL-System erfordert weiteren Entwicklungsaufwand, der im Rahmen dieser Diplomarbeit nicht möglich war. Dennoch zeichnet sich ab, daß der Grundalgorithmus stabil ist und eine evolutionäre Weiterentwicklung möglich ist.

Im einzelnen wurden realisert:

- Eine syntaktische Analyse, aufbauend auf hierarchischen Automaten, wurde erstellt, die im wesentlichen Nominal-, Präpositionalphrasen und Satzgrenzen erkennt. Bei syntaktisch nicht zu komplizierten Sätzen arbeitet sie bereits zufriedenstellend.

- Ein Translation-Template-Generierungs-Modul, das aus einem vorhandenen bilingualen VERBMOBIL-Korpus Translation Templates extrahieren kann. Dabei wurde der HTG-Ansatz so erweitert, daß die Part-of-Speech und die Satzstruktur beim Alignment stärker berücksichtigt werden. Allerdings hat sich dabei gezeigt, daß auf eine manuelle Nachbearbeitung nicht verzichtet werden kann. Trotzdem ist dies effizienter, als die Translation Templates manuell zu erstellen.

- Ein Transfermodul, das zum einen die Auswahl bei mehreren alternativen Translation Templates vornimmt, zum anderen die einzelnen zielsprachlichen Templates zu einem Synatxbaum zusammenfügt. Bei der Auswahl der Templates wird dabei auf eine manuell erstellte Gewichtung zurückgegriffen, die recht gut funktioniert. Durch die eingesetzten Optimierungen ist das Modul relativ schnell.

- Eine Generierungskomponente, die innerhalb der Knoten mittels hierarchischer Automaten die Wörter richtig ordnet. Die Grundstruktur stimmt in den meisten Fällen, die Behandlung von einzelnen Problemen, z.B. die Stellung von Adverbien, ist allerdings noch zu verbessern.

Der geradlinige, relativ einfache Aufbau des Systems hat sich bewährt. Bisher konnten alle Erweiterungen ohne größere Änderungen an der grundlegenden Struktur des Programms vorgenommen werden. Insbesondere die von dem ausgangs- und zielsprachlichen Syntaxbaum relativ unabhängige Übersetzung ermöglicht eine unkomplizierte Verarbeitung.

Das Ziel wiederverwertbare Module zu entwickeln, scheint erreicht worden zu sein. So konnte der Automat schon in dem Tool zur Zeitextraktion eingesetzt werden, wobei speziell die Möglichkeit Chunks zu generieren, genutzt wird. Ein weiteres Gebiet ist der Aufbau von semantischen Beschreibungen aus natürlichsprachlichen Sätzen. Für das Transfermodul zeichnen sich ebenfalls bereits Aufgabenfelder ab, z.B. Dialogakt-basierte Übersetzung und Protokollgenerierung.

9.2 Ausblick

Durch die vorgenommene Evaluation läßt sich recht gut abschätzen, in welche Richtung das System verbessert bzw. erweitert werden kann.

Für das eigentliche Übersetzungsmodul erscheinen folgende Punkte am dringlichsten:

- Einbau einer semantischen Wortdistanzfunktion. Dabei kann auf eine im Rahmen des VERBMOBIL-Projekts erstellte semantische Datenbank zurückgeriffen werden.
- Mit Hilfe der dort gespeicherten Informationen kann auch die syntaktische Analyse erweitert bzw. verbessert werden, so z.B. PP-Attachment, eine verbesserte Satzgrenzenerkennung, Abgrenzung von Infinitivsätzen.
- starke Erweiterung des Beispielkorpus und manuelle Überarbeitung der erstellten Translation Templates
- Bei der Auswahl des Translation Templates sollte der zielsprachliche Teil berücksichtigt werden
- Die Generierungsfunktion sollte allgemein verbessert und um ein morphologisches Modul erweitert werden.
- Berücksichtigung der Zeit beim Übersetzen von Verben
- Deutsch-Englische Übersetzungsrichtung

Als weniger dringlich stellen sich hingegen folgende Punkte dar (zumindest bezogen auf den verwendeten Korpus):

- Verbesserung der Gewichtungsfunktionsfunktion, diese scheint recht zufriedenstellend zu funktionieren

- Morphologische Analyse des Ausgangssatzes zur Subjekt- bzw. Objektbestimmung

Für die Alignmentkomponente ergeben sich als Erweiterungsmöglichkeit noch die inkrementelle Ergänzung des Wörterbuchs. Ein vollautomatisches Alignment scheint den Evaluationsergebnissen nach nicht machbar zu sein. Der Schwerpunkt sollte daher auf einer Funktion liegen, die ermitteln kann, ob beim Alignment ein Fehler aufgetreten sein könnte und so die manuelle Nachbearbeitung zu erleichtern.

In Hinblick auf den Einsatz in einem System mit fehlerbehafteter natürlichsprachlicher Eingabe ergeben sich ganz neue Herausforderungen:

- Einsatz des Beispielsatzkorpus, um aus mehreren möglichen erkannten Wortfolgen, die am besten zu dem Korpus passende zu finden.
- Verstärkter Einsatz von vorgefertigten Mustersätzen, um zu verhindern, daß Fehler bei der Erkennung zu unverständlichen Übersetzungsergebnissen führen. Dafür kann das vorhandene Übersetzungsmodul ohne aufwendige Änderungen eingesetzt werden.

Anhang A

Dateiformate

Definition der hierarchischen Automaten

Anhand eines Beispiels soll der typische Aufbau einer Datei, die einen hierarchischen Automaten definiert, gezeigt werden:

```
#! begin automaton level1.sublevel1.automat1
#! begin definition
.r S1
.f S2
.f S3
a S1 S2 b
c S1 S3 d
#! end definition
#! end automaton
#! begin automaton level1.sublevel2.automat1
#! begin definition
...
```

Bei dem gezeigten Automaten sind drei Knoten definiert: *S1*,*S2* und *S3*, zudem zwei Übergänge, wobei der Übergang zwischen *S1* und *S2* die Bedingung `a` und die Aktion `b`, der Übergang zwischen *S1* und *S3* die Bedingung `c` und die Aktion `d` hat. *S1* wird mit `.r` zum Startzustand definiert, *S2* und *S3* werden durch `.f` als Endzustände definiert.

Die einzelnen Automaten können einfach untereinander geschrieben werden, die hierarchische Struktur wird durch den Namen festgelegt. In dem Beispiel werden zwei Level angelegt, einer für *level1.sublevel1* und einer für *level1.sublevel2*.

Tagged-Format

Dieses Format wird von den POS-Tagger und der Zeiterkennungskomponente als Ausgabe verwendet und von den hierarischen Automaten der syntaktischen Analyse eingelesen.

In einer Zeile stehen dabei das eigentliche Wort, die POS des Wortes und die Stammform, jeweils durch Leerzeichen getrennt. Satzendezeichen ist die Folge

```
. $. .
```

Mehrere Sätze können dabei aneinandergehängt werden.

Chunked-Format

Ausgabe der syntaktischen Komponente erfolgt in diesem Format. Der Syntaxbaum ist dabei als ineinander geschachtelte Chunks kodiert.

Chunks werden in folgender Form dargestellt

```
[ <Name> <Chunk oder Wort> ]
```

und Wörter in dieser Form

```
[ <Wort> <POS> <Wortstamm> ]
```

Chunked-Pair-Format

Eingabe des Template-Generierungstools erfolgt in diesem Format. Die Dateien bestehen aus folgenden Einträgen, die aneinander gehängt werden können:

```
<Index>
<deutscher Syntaxbaum als geschachtelte Chunk-Struktur>
<englischer Syntaxbaum als geschachtelte Chunk-Struktur>
```

Aligned-Pair-Format

Ausgabe der Template-Generierungskomponente und Eingabe für das Übersetzungsmodul. Der offline alignierte Beispielkorpus wird also in diesem Format gespeichert.

Das Format ist identisch zu dem Chunk-Pair-Format, allerdings sind innerhalb der Chunks bzw. der Wörter Link-Informationen zusätzlich angegeben. Chunks sehen folgendermaßen aus:

```
[ <Name> <Link-Information> <Chunk oder Wort>]
```

und Wörter so:

```
[ <Wort> <POS> <Stammform> <Link-Information>]
```

Anhang B

Beispiele

Die Satzpaare des Korpus 2 (`ger`,`eng`) sind hier zusammen mit den Übersetzungsresultaten (`chunky`) vollständig aufgelistet.

```
c2007
ger : haben Sie sich denn schon mal informiert .
eng : have you found out anything at all .
chunky : do you have you already informed ?

c2009
ger : hallo Peter gut dass wir uns treffen .
eng : hello Peter it is good that we are meeting .
chunky : hello Peter that we meet great

c2024
ger : ich wuerde sagen Flugzeug ist doch angenehmer .
eng : I would say a plane is indeed more convenient .
chunky : I would say the plane would be more convenient for

c2026
ger : in den naechsten Tagen sieht es schwierig aus .
eng : it looks difficult in the next few days .
chunky : in the next it look difficult next day

c2029
ger : insofern waere das Abendprogramm eigentlich auch schon etwas .
eng : as far as that is concerned the program for the evening is already
      somewhat .
chunky : we could still have dinner or something like that

c2036
ger : ja da finden wir dann sicher was .
eng : yes we will definitely find something there then .
chunky : yes there can certainly find what then

c2039
ger : ja dann fahren wir mit dem Zug .
eng : well then let us go by train .
chunky : certainly then let us go by train
```

```
c2041
ger : ja dann haben wir uns ja geeinigt .
eng : yes then we have reached an agreement .
chunky : then has well we agree

c2055
ger : ja das koennen wir ja dann sehen .
eng : yes well we can see that then .
chunky : well we can see then

c2057
ger : ja das waere mir auch ganz lieb .
eng : yes I would like that too .
chunky : yes I would like it too

c2087
ger : ja weil wir eineinhalb Tage bleiben wollen .
eng : yes because we want to stay a day and a half .
chunky : because yes we want to stay eineinhalb_ger a

c2092
ger : ja eben also machen wir das dann .
eng : yes exactly well let us do that then .
chunky : yes let us just so do it then

c2096
ger : kann ich das direkt bei Ihnen festmachen .
eng : can I arrange that directly with you .
chunky : can I arrange that right for you ?

c2124
ger : so wie wollen wir denn hinkommen .
eng : well then how do we want to get there .
chunky : around how do we want to get there then ?

c2136
ger : und ich reserviere dann da zwei Plaetze vor .
eng : and then I will book two seats there in advance .
chunky : and then I vorreserviere_ger seat two

c2137
ger : und was sollte man da noch besprechen .
eng : and what else should we discuss here .
chunky : oh that means Maritim that advance and the court

c2140
ger : und wo wollen wir uns dann treffen .
eng : and where do we want to meet then .
chunky : and where do we want to meet then ?

c2166
ger : wie sieht es denn aus im Maerz .
eng : how does it look then in March .
chunky : how does it look then at March ?
```

c2170
ger : wie waere es denn gleich Anfang Juni .
eng : how about right at the beginning of June then .
chunky : how it will be in no time at the beginning of June ?

c2179
ger : wollen wir irgendwo etwas unternehmen oder nicht .
eng : do we want to do something somewhere or not .
chunky : do we somewhere want to anything or right ?

c2183
ger : Donnerstag spaeter abends wieder zurueck nicht wahr .
eng : back again late Thursday evening right .
chunky : back Thursday late in in the evening back right true

c2186
ger : Reiseagentur Maritim Fischenbeck am Apparat guten Tag .
eng : travel agency Maritim Fischenbeck speaking hello .
chunky : do Maritim Reiseagentur_ger good Fischenbeck is talking day ?

c2196
ger : aber ist eigentlich schlecht am Wochenende oder .
eng : but it is actually bad on the weekend right .
chunky : but is at the weekend bad actually or ?

c2205
ger : also da habe ich ganz viel an Dienstbesprechungen .
eng : well I have got so many planning meetings there .
chunky : then well I completely do a lot of in Dienstbesprechungen_ger

c2209
ger : also im Januar wuerde es mir gut passen .
eng : well it would suit me fine in January .
chunky : at well it would suit me well January

c2233
ger : dann Unterkunft brauchen wir ja natuerlich auch noch .
eng : then certainly we also need accommodation of course .
chunky : then we certainly also need accommodation of course

c2243
ger : dann ist denke ich soweit alles klar .
eng : then everything I think is alright so far .
chunky : then I think is cleared up so far actually everything

c2248
ger : dann schauen wir mal wo wir schlafen koennen .
eng : then let us have a look where we can sleep .
chunky : then let us take a look where we can nod

c2257
ger : das ist der elfte zwoelfte ne .
eng : that is the eleventh twelfth right .
chunky : hat is eleventh or twelfth right

```
c2268
ger : die Reise soll drei bis vier Tage dauern .
eng : the trip should last three to four days .
chunky : the trip should three

c2281
ger : gut dann buche ich da zwei Einzelzimmer vor .
eng : great then I will book two single rooms there in advance .
chunky : good then I will book two single rooms for the Luisenhof there

c2295
ger : hoert sich auch eigentlich ganz gut an ne .
eng : it also sounds quite good actually right .
chunky : do also actually quite sounds okay good ?

c2305
ger : ich muss im Computer gucken ob das geht .
eng : I have to take a look in the computer if that is possible .
chunky : I have to take a look in the computer if that works

c2320
ger : ist Ihnen der um sieben Uhr zu frueh .
eng : is the one at seven o'clock too early for you .
chunky : you is it seven oh-four in the morning ?

c2340
ger : ja das glaube ich waere sinnvoll .
eng : yes I think that would be sensible .
chunky : yes I think is suggestive that

c2344
ger : ja eigentlich ist doch soweit alles jetzt geklaert .
eng : yes everything certainly seems to be settled so far for now .
chunky : actually everything will large now cleared

c2364
ger : ja natuerlich den kann ich Ihnen buchen .
eng : yes of course I can book that for you .
chunky : yes of course this I can book you

c2402
ger : okay das kann man ja dann noch entscheiden .
eng : okay well that can still be decided on then .
chunky : you still can decide that yes okay

c2405
ger : sagen Sie mir Ihren Namen noch mal bitte .
eng : would you tell me your name once more please .
chunky : do you please should me say your name ?

c2415
ger : und was machen wir denn am Abend .
eng : and what shall we do in the evening .
chunky : and what we do in the evening then
```

```
c2467
ger : Sie haben recht bisschen ausschlafen ist auch nicht schlecht .
eng : you are right sleeping in a little is also not bad .
chunky : have not you sleep late got a bit too bad right is

c2491
ger : bei mir steht da naemlich auch schon eine Dienstreise drin .
eng : there is also already a business trip in there for me then .
chunky : for me a work trip given too too there inside

c2497
ger : da gibt es das Maritim-Hotel in der Naehe vom Hauptbahnhof .
eng : there is the Maritim hotel near the main train station .
chunky : there the Maritim-Hotel they have near the airport from the
         main  train station

c2498
ger : da habe ich bis jetzt noch nichts in meinem Terminkalender .
eng : I do not have anything there yet in my calendar .
chunky : then I still do not have anything in my calendar until now

c2505
ger : dann frage ich mal an was da so laeuft .
eng : then I will just ask what is playing there .
chunky : then I just anfrage_ger what so is running

c2508
ger : dann machen wir das doch gleich fest ne .
eng : then let us just settle that right away okay .
chunky : we just arrange sure that then

c2520
ger : das sind hundert neun-zehn Mark das sollte wohl hinkommen .
eng : that is one-hundred-nineteen marks that should be feasible .
chunky : that is one-hundred-twenty-nine marks ten hundred that
         probably should get

c2523
ger : dem Namen nach ich bin dort noch nie gewesen .
eng : by name I have never been there .
chunky : the name after I have never been there

c2542
ger : haben Sie da im Zentrum verschiedene Hotels zur Auswahl .
eng : do you have several hotels in the city center there to choose from .
chunky : do you there at center have here several hotels to choose from ?

c2561
ger : ich wuerde sagen wir fahren am Vormittag irgendwann .
eng : I would say we go sometime in the morning .
chunky : I would say let us sometime go in the morning until

c2564
ger : ja Rueckflug muss ich ja dann auch gleich buchen .
eng : yes I have to book the return trip right away then too .
chunky : right then I will ahead also have to book flight
```

```
c2569
ger : ja da kann ich Ihnen also ein Zimmer anbieten .
eng : yes well I can offer you a room there .
chunky : yes well I can offer a rooms

c2577
ger : ja dann waere es das eigentlich schon ne .
eng : well actually that was it already right .
chunky : well well it really is it then right

c2578
ger : ja dann war es das eigentlich denke ich .
eng : yes then that was it actually I think .
chunky : then well well it really is that I think

c2582
ger : ja das Hotel Cristal ist im allgemeinen ganz gut .
eng : yes the hotel Cristal is generally very good .
chunky : yes the hotel Cristal_ger is very good at abstract

c2584
ger : ja das ist bei mir auch noch alles frei .
eng : well that is also all still free for me .
chunky: yes that still is also open for me everything

c2599
ger : ja ich denke das liesse sich machen ja .
eng : yes I think that could be arranged yes .
chunky : yes I think yes why do not let they do do that

c2605
ger : ja lassen Sie uns das nicht so knapp veranschlagen .
eng : well let us not cut it so tight .
chunky : yes you mhm do not estimate us let tight that

c2611
ger : ja wann fahren wir da am besten los .
eng : well what time should we leave .
chunky : when do we preferably leave yes there ?

c2618
ger : ja wunderbar daran habe ich auch schon gedacht .
eng : well wonderful I have also already thought about that .
chunky : yes I also already think so do wonderful at it

c2623
ger : jetzt muessen wir noch ueberlegen wie wir da hinkommen .
eng : now we have to think about how we will get there .
chunky : now we still have to think how we will get there

c2643
ger : und der Preis fuer das Einzelzimmer betraegt hundert
      fuenf-und-dreissig Mark .
eng : and the price for a single room is one-hundred-thirty-five marks .
chunky : and the price amounts one-hundred-twenty-nine marks for the
         single room hundred 35
```

c2659
ger : wann geht denn der zweite Zug in der Frueh .
eng : what time does the second train in the morning go then .
chunky : when do train goes then second ?

c2672
ger : wir koennten noch Essen gehen oder irgendwie sowas .
eng : we could still have dinner or something like that .
chunky : or we still somehow could eat sowas_ger

c2673
ger : wollen Sie denn fliegen oder mit dem Zug fahren .
eng : do you want to fly then or go by train .
chunky : did do you want to go then or go fly go by train ?

c2683
ger : ach das heisst Maritim das Luisenhof und der Loccumer-Hof .
eng : oh they are called Maritim the Luisenhof und the Loccumer-Hof .
chunky : oh that means Maritim that advance and the court

c2704
ger : bezueglich des Verkehrsmittels was haben Sie sich da vorgestellt .
eng : as far as transportation is concerned what did you have in mind there .
chunky : regarding the Verkehrsmittels_ger what did have in mind for that

c2721
ger : das waere dann gleich die erste nach Hannover von Muenchen aus .
eng : that is then the very first to Hannover from Munich .
chunky : then it just auswaere_ger to Hannover from Munich first

c2729
ger : die Fahrtdauer betraegt dann drei Stunden zwanzig Minuten aber das .
eng : the duration of the trip is three hours and twenty minutes then but
      that .
chunky : then the pm amounts that three hours twenty minutes but

c2781
ger : ja ich kann nur noch nicht in der ersten Oktoberhaelfte .
eng : well only I can not make it in the first half of October .
chunky : yes I could not yet only make it first Oktoberhaelfte_ger

c2782
ger : ja ich wuerde dann wohl den um sieben fuenf-und-fuenfzig nehmen .
eng : well I guess I will take the one at seven fifty-five am then .
chunky : yes I would probably take fifty after seven fourty-five am then

c2788
ger : ja nee dann das waere es vorerst von mir aus .
eng : well no then that would be it from my side for the time being .
chunky : no then yes it vorerst_ger auswaere_ger that from me

c2801
ger : jetzt brauchen wir natuerlich noch ein Hotel wuerde ich sagen .
eng : and now we still need a hotel of course I would say .
chunky : now we need a hotel of course I would say

```
c2804
ger : man kann ja dann erster Klasse auch reisen oder so .
eng : one can certainly also travel first class then or something .
chunky : well you also can travel first class then or so

c2805
ger : mhm dann bis Donnerstag den drei-und-zwanzigsten Juli in Hannover .
eng : mhm then until Thursday the twenty-third of Juli in Hannover .
chunky : to twenty third Thursday July mhm then in Hannover

c2807
ger : na dann sollten wir uns noch um das Hotel kuemmern .
eng : well then we should still take care of a hotel .
chunky : then well we still should kuemmern_ger us of the hotel

c2812
ger : nehmen wir den sind wir eine Stunde spaeter in Koeln-Bonn .
eng : let us take it we will be an hour later in Cologne-Bonn .
chunky : let do us take it later is we one hours at Cologne-Bonn ?

c2825
ger : und mit der Zugverbindung wie lange wird das dauern .
eng : and as far as the train connection is concerned how long will that take .
chunky : and with the train connections how will it takes long ?

c2828
ger : vom elften bis zum nein vom zwoelften bis zum fuenf-zehnten .
eng : from the eleventh until the no from the twelfth until the fifteenth .
chunky : after eleventh until for no between twelfth and fifteenth

c2835
ger : wir treffen uns um eine Geschaeftsreise nach Hannover zu vereinbaren .
eng : we are meeting in order to arrange a business trip to Hannover .
chunky : we will meet to arrange a business trip to Hannover

c2853
ger : also Hannover hat glaube ich schon ein einigermassen interessantes
      Nachtleben .
eng : well Hannover does have I think a somewhat interesting night life .
chunky : well Hannover certainly have a to some extent interessantes_ger
         Nachtleben_ger I think

c2861
ger : bis wir dann in der Stadt sind noch mal vierzig Minuten dran .
eng : then until we are in the city another forty minutes more .
chunky : until then we just dransind_ger in the city

c2865
ger : da ist auch ein Konferenzraum falls noch etwas zu besprechen ist .
eng : there is also a conference room in case there is still something to
      discuss .
chunky : a the conference room will also if anything to discuss is
```

```
c2870
ger : dann fragt sich jetzt nur noch ob wir abends miteinander ausgehen .
eng : then now there is just the question of whether we go out together in
      the evening .
chunky : asks then you only together go out we now if in the evening

c2884
ger : das kostet das Einzelzimmer hundert fuenf-und-zwanzig Mark und
      das Doppelzimmer sechs-und-neunzig Mark .
eng : it costs one-hundred-twenty-five per single room and ninety-six marks per
      double room .
chunky : it the hundred eight $p $m two-hundred-fourty-eight marks and the
         single room double room one-hundred-eighty-three 96
         two-hundred-fourty-eight marks

c2894
ger : der faehrt ah Entschuldigung der faehrt nur Samstag Sonntag .
eng : it goes oh sorry it goes only Saturdays and Sundays .
chunky : it departs oh apology it only departs Saturday or Sunday

c2898
ger : eines der Hotels ist das Grand-Hotel Mussmann das liegt sehr zentral .
eng : one of the hotels is the Grand Hotel Mussmann that is very centrally
      located .
chunky : the Grand-Hotel_ger Mussmann_ger is ones the hotels it is very central

c2903
ger : es wird doch ein ziemlich langer Tag meinen Sie nicht .
eng : it will certainly be quite a long day do you not think .
chunky : it would a quite long days week you meant right

c2911
ger : haben Sie sich schon schlau gemacht wann Sie fahren koennten .
eng : have you already found out when you could go .
chunky : why do you do not already have make cleverly you when you could go ?

c2913
ger : ich denke das entscheiden wir wenn wir dann da sind .
eng : I think let us decide that when we are there then .
chunky : I think we will decide that if we will there then

c2916
ger : ich hoffe auch nicht dass das noetig sein wird ja .
eng : I also hope that that will not be necessary yes .
chunky : I do not hope too that it would be neccessary yes

c2928
ger : ja Zuege fahren halt stuendlich ab sechs fuenf-und-vierzig bis
      acht-zehn fuenf-und-vierzig .
eng : yes there are trains going hourly from six-fourty-five am till
      six-fourty-five pm actually .
chunky : train goes hourly yes between forty five after six and forty five
         after ten
```

```
c2929
ger : ja aber gehen wir doch dann Sonntag abend noch ins Theater .
eng : well but let us still go to the theater then on Sunday evening .
chunky : well let do us go to the theater then but Sunday in the evening ?

c2952
ger : ja schoenen guten Tag Frau Klentzky Berg hier .
eng : well a hearty hello Mrs Klentzky Berg here .
chunky : well a hearty hello Mrs Klentzky mount here

c2968
ger : oh ja dann koennte ich am Freitag auch schon losfahren .
eng : oh well then I could leave on Friday already too .
chunky : oh then well I also certainly could leave early Friday

c2970
ger : okay dann werde ich mal da fuer Sie ein Ticket reservieren .
eng : okay then I will go ahead and book a ticket for you then .
chunky : okay then I will go ahead and some tickets there for you

c2973
ger : soll ich Ihnen denn schon mal im Hotel ein Zimmer reservieren .
eng : so shall I book a room for you in the hotel then .
chunky : should I book you then at the hotel a rooms ?

c2979
ger : tja muessen wir gucken dass wir vier aufeinanderfolgende Tage
      zusammenbekommen .
eng : well we have to see that we will get four succesive days together .
chunky : well that we have to take a look we four successive days get together

c2998
ger : zu welchem Datum oder welchem Monat waere es Ihnen denn genehm .
eng : at which date or in which month would it be okay with you then .
chunky : at it be genehm_ger which date or which month you

c3000
ger : zwei-hundert dreissig Mark und eine einfache Fahrt zurueck ja .
eng : two-hundred-thirty marks and an one way trip back yes .
chunky : two-hundred-thirty marks a simplemindedly ride and back yes
```

Literaturverzeichnis

[ABH+94] D. Arnold, L. Balkan, R. Lee Humphreys, S. Meijer, and L. Sadler. *Machine Translation – An Introductory Guide*. NCC Blackwell, 1994.

[Abn91] Steven Abney. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle Based Parsing*. Kluwer Academic Publishers, 1991.

[Abn96] Steven Abney. Partial parsing via finite-state cascades. In *Proceedings of th ESSLLI '96 Robust Parsing Workshop*, 1996.

[ASJ90] Anne Abeillé, Yves Schabes, and Aravind K. Joshi. Using lexicalized tags for machine translation. In *Proc. of Coling '90*, Helsinki, Finland, 1990.

[Bea92] J.L. Beaven. Shake-and-bake translation. In *Proc. of Coling '92)*, pages 603–609, 1992.

[BKW96] Christof Burgard, Reinhard Karger, and Wolfgang Wahlster. Wissenschaftliche Ziele und Netzpläne für Verbmobil Phase 2. Technisches Dokument 44, DFKI GmbH, 1996.

[Bro96] Ralf D. Brown. Example-based machine translation in the pangloss system. In *Proc. of Coling '96*, pages 169–174, 1996.

[Deu98a] *Deutsch-englisches Wörterbuch.* `http://ftp.leo.org/pub/comp/doc/dict/eng2ger.vok.gz`, 1998.

[Deu98b] *Deutsch-englisches Wörterbuch.* `http://ftp.fu-berlin.de/pub/misc/dictionaries/book.vok.gz`, 1998.

[Edw97] Stephen Edwards. *States - An Automata Drawing Tool.* `http://www.eecs.berkeley.edu/~sedwards`, 1997.

[End98] Ulrich Endriss. Semantik zeitlicher Ausdrücke in Terminvereinbarungsdialogen. Verbmobil-Report 227, Technische Universität Berlin, August 1998.

[Eva97] Edmund Grimley Evans. Approximating context-free grammars with a finate-state calcus. In *ACL '97*, pages 452–459, 1997.

[Fas98] *Fastus.* `http://www.ai.sri.com/natural-language/projects/fastus`, 1998.

[FI92a] Osamu Furuse and Hitoshi Iida. Cooperation between transfer and analysis in example-based framework. In *Proc. of Coling '92*, pages 645–651, 1992.

[FI92b] Osamu Furuse and Hitoshi Iida. An example-based method for transfer-driven machine translation. In *TMI '92*, pages 139–150, 1992.

[FI94] Osamu Furuse and Hitoshi Iida. Constituent boundary parsing for example-based machine translation. In *Proc. of Coling '94*, pages 105–111, 1994.

[FI96] Osamu Furuse and Hitoshi Iida. Incremental translation utilizing constituent boundary patterns. In *Proc. of Coling '96*, pages 412–417, 1996.

[HM97] Ulf Hermjakob and Raymond J. Mooney. Learning parse and translation decisions from examples with rich context. In *ACL '97*, pages 482–489, 1997.

[HS92] W. John Hutchins and Harold S. Somers. *An Introduction to Machine Translation.* Academic Press, 1992.

[ISF96] Hitoshi Iida, Eiichiro Sumita, and Osamu Furuse. Spoken-language translation method using examples. In *Proc. of Coling '96*, pages 1074–1077, 1996.

[KKM92] Hiroyuki Kaji, Yuuko Kida, and Yasutsugu Morimoto. Learning translation templates from bilingual text. In *Proc. of Coling '92*, pages 672–678, 1992.

[MW92] Hiroshi Maruyama and Hideo Watanabe. Tree cover search algorithm for example-based translation. In *TMI '92*, pages 173–184, 1992.

[NBD94] Sergei Nirenburg, Stephen Beale, and Constantine Domashev. A full-text expeiment in example-based machine translation. In *New Methods in Language Processing*, 1994.

[NCTG92] Sergei Nirenburg, Jaime Carbonell, Masaru Tomita, and Kenneth Goodman. *Machine Translation – A Knowledge-Based Approach.* Morgan Kaufmann Publishers, 1992.

[Nir95] Sergei Nirenburg. The pangloss mark iii machine translation system. Technical Report CMU-CMT-95-145, Computing Research Laboratory (New Mexico State Univeristy, Center for Machine Translation (Carnegie Mellon University), Information Sciences Institute (University of Southern California, 1995.

[Nom92] Hiroshi Nomiyama. Machine translation by case generalization. In *Proc. of Coling '92*, pages 714–721, 1992.

[POS98] *Part-of-Speech Tagger fürs Deutsche.* `http://www.ims.uni-stuttgart.de/`, 1998.

[Pri97] Gilles Prigent. Synchronous tree adjoining grammars. In `http://www.cis.upenn.edu/~xtag/stag/`, 1997.

[San95] Beatrice Santorini. *Part-of-Speech Tagging Guidelines for the Penn Treebank.* `http://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz`, 1995.

[Sen98] Jean Senellart. Locating noun phrases with finate state transducers. In *Proc. of Coling '98*, pages 1212–1218, 1998.

[SI91] Eiichiro Sumita and Hitoshi Iida. Experiments ans prospects of example-based machine translation. In *ACL '91*, pages 185–192, 1991.

[SI95] Eiichiro Sumita and Hitoshi Iida. Heterogeneous computing for example-based translation of spoken language. In *TMI '95*, pages 273–285, 1995.

[SN90] Satoshi Sato and Makoto Nagao. Toward memory-based translation. In *Proc. of Coling '90*, pages 247–252, 1990.

[SS90] Yves Schabes and Stuart Shieber. Synchronous tree adjoining grammars. In *Proc. of Coling '90*, 1990.

[STST95] Anne Schiller, Simone Teufel, Christine Stückert, and Christine Thielen. *Vorläufige Guidelines für das Tagging deutscher Textcorpora mit STTS.* `http://www.ims.uni-stuttgart.de/ftp/pub/corpora/stts_guide.ps.gz`, 1995.

[Tag98] *Rule Based Tagger for English Part-of-Speech.* `http://www.de.relator.research.org/resources/rule_based_tagger`, Nov. 1998.

[Wah93] Wolfgang Wahlster. Verbmobil–Translation of Face-to-Face Dialogs. Technical report, German Research Centre for Artificial Intelligence (DFKI), 1993. In Proceedings of MT Summit IV, Kobe, Japan, 1993.

[Wat94] Hideo Watanabe. A method for distinguishing execptional and general examples in example-based transfer systems. In *Proc. of Coling '94*, pages 39–43, 1994.

[Whi92] Peter Whitelock. Shake-and-bake translation. In *Proc. of Coling '92*, pages 784–790, 1992.

[ZW98] Klaus Zechner and Alex Waibel. Using chunk based partial parsing of spontaneous speech in unrestricted domains for reducing word error rate in speech recognition. In *Proc. of Coling '98*, pages 1453–1459, 1998.