Verstehen natürlicher Sprache
durch statistische Übersetzung
in eine termbasierte
Interlingua

Diplomarbeit
von
Manuel Kauers

betreut von
Prof. Dr. rer. nat. Alex Waibel
Dipl. Phys. Stephan Vogel
Dipl.-Inform. Christian Fügen

Mai 2002
Zusammenfassung

Die vorliegende Diplomarbeit beschäftigt sich mit dem Problem des automatischen „Verstehens“ von natürlicher Sprache. Im Zusammenhang mit automatisierten Systemen versteht man darunter die Generierung einer sprachunabhängigen Darstellung der Semantik eines natürlichsprachlich gegebenen Textes.


Die gewonnenen Erkenntnisse wurden zu Evaluationszwecken in einem Softwaresystem implementiert, so daß es möglich war, verschiedene Verfahren zu testen. Dabei zeigte sich, daß trotz der vergleichsweisen Einfachheit des Ansatzes und trotz nur spärlich vorhandener Trainingsdaten überraschend gute Ergebnisse erzielt werden können. Gleichzeitig werden durch die Ergebnisse der Experimente neue Fragstellungen aufgeworfen, die interessante Möglichkeiten zu weiteren Forschungen aufzeigen.
Zusammenfassung
Abstract

This diploma thesis considers the problem of natural language understanding. In the context of computer systems, this means the process of transferring an utterance given in natural language into a suitable language independent representation of its semantic content.

In contrast to state-of-the-art techniques which do mainly some kind of rule based analysis, the method proposed in this thesis is an adaptation of techniques known from statistical machine translation. Instead of applying those techniques to a pair of two natural languages, e.g. to translate from German to English, we consider the intermediate representation as the target language of a statistical translator. Natural language understanding is thus regarded as "translation" from a natural source language into this target language. As most of today's intermediate languages (especially the Interchange Format used at our institute) have a tree-like structure while classical translation models regard phrases as linear chains of words, a major part of the work is dedicated to the necessary generalization of existing statistical models and algorithms to objects of that particular form.

A great advantage of automatically trainable systems over handcrafted ones is of economic nature: An existing statistical system is able to learn its parameters within minutes and provides results of a quality which is comparable to the quality of results obtained by grammar based systems with grammars that have taken a great effort to build. However, this calculation does not take into account the need of a large bilingual corpus. Since we need for our situation a corpus which maps utterances to corresponding Interchange Format expressions, we still need experts to compile a corpus file. This is why we do also propose a method which allows for extracting the needed parameters from a bilingual corpus of two natural languages. The idea is to obtain a model for, say, German to Interchange Format given a model for German to English and another model for English to Interchange Format.

Our ideas have been implemented for evaluation purposes to allow for a comparison to established systems. Experiments carried out with that implementation showed that though our approach is rather simple and though our training set is not very large, the results we obtain are of surprisingly high quality. At the same time, the results of our experiments point out new open problems and interesting directions for further research.
Résumé

Le présent mémoire traite du problème de compréhension automatique d’une langue naturelle. En terme de systèmes automatisés cela signifie le transfert d’un texte en langue naturelle à une représentation de la même sémantique qui est indépendante de la langue.

A la différence des procédés conventionnels généralement fondés sur des techniques d’analyse des règles, le procédé développé dans ce mémoire adapte plusieurs méthodes utilisées dans la traduction automatique et statistique. Au lieu d’appliquer ces techniques habituelles à une paire de langues naturelles, par exemple pour traduire de l’allemand vers l’anglais, la représentation intermédiaire est considérée comme étant la langue cible d’un traducteur statistique. Notre étude décrit également la compréhension d’une langue naturelle comme « traduction » d’une langue source naturelle en cette langue cible. Comme la plupart de ces langages intermédiaires (en particulier le format Interchange Format utilisé par notre institut) ont une structure similaire à celle d’une arborescence alors que les modèles de traduction classiques interprètent des phrases comme des chaînes linéaires de mots, cette étude sur la généralisation des modèles statistiques existants et des algorithmes se consacre en grande partie aux objets de cette forme particulière.

L’avantage majeur des systèmes automatisés par rapport aux systèmes manuels est d’ordre économique. Ainsi, un système statistique, une fois implémenté, est en mesure d’apprendre des paramètres en quelques minutes et de livrer des résultats d’une qualité comparable à la qualité des résultats obtenus par des systèmes grammaticaux utilisant des règles de grammaire créées manuellement. Toutefois, ce procédé ne tient pas compte du besoin d’un glossaire bilingue. Etant donné que dans notre cas un glossaire est nécessaire pour faire correspondre les expressions de la langue naturelles aux expressions au format Interchange, nous devons toujours faire appel à des terminologies pour créer un fichier glossaire. Pour cette raison nous proposons une méthode qui permet d’extraire les paramètres nécessaires à la création d’un glossaire de deux langues naturelles. L’idée est d’obtenir un modèle – par exemple – l’allemand vers le format Interchange à partir de deux modèles déjà existants : l’allemand vers l’anglais et l’anglais vers le format Interchange.

Nos idées ont été développées sous forme de logiciel de manière à pouvoir comparer ce système aux systèmes existants et ainsi évaluer les nouvelles connaissances. Bien qu’il n’y ait pas eu de découvertes fondamentales dans nos recherches et que nous ne disposions que de très peu de données de test, les expériences ont toutefois donné des résultats surprenants. De plus, les résultats des expériences soulèvent des problèmes non résolus donnant ainsi de nouvelles directions aux futures recherches.
Resumo

Tiu ĉi diploma laboro okupiĝas pri la problemo de la aŭtomata kompreno de naturlingvo. En la kuneksto de komputilsistemoj tio signifas kreadon de lingvosependa reprezenton de la sektanto de naturlingva teksto.

La metodo, kiun elvolvatas en ĉi-laboro estas – alimaniere ol la kutime uzata, kiuj ĉefe uzas regulbazitajn analizteknikojn – adaptigo de metodoj uzataj je statistika aŭtomata traduko. Anstataŭ aplikado de tiuj metodoj al paro de naturaj lingvoj por ekz. traduko de la Germana al la Angla, ni konfirma la interlingvan reprezenton kvazaŭ cellingvo de la statistika traduksystemo kaj la lingvokomprono kvazaŭ traduko de naturlingva foutclingvo al tiu cellingvo. La klasikaj tradukmodeloj konsideras frazojn kiel linearaj ĉenoj de vortoj, sed la plejparto de la kutimaj interlingvoj – precipe la Interchange Format uzata en nia instituto – havas arban strukturon; pro tio la centra parto de ĉi-laboro okupiĝas pri la neceso ĝeneraligo de ekzistantaj statistikaj modeloj kaj algoritmoj al tiuj strukturoj.

Ĉefa avantajo de aŭtomate trejnblaj metodoj kompare kun mane kreitaj regul sistemoj estas ekonomia: Statistika sistemo povas – unuoje programita – dum kelkaj minutoj lerni parametrojn, kun kiuj eblas ricevi rezultojn, kiuj estas rilate al la kvalito kompareblaj kun tiuj de sistemoj uzataj gramatikojn – mankcreitajn per multe la laboro. Sed je tio kalkulo oni ne pripensas, ke por la trejnado de la parametroj necesas kiel eble plej granda tekstkorpuso. Ĉar jena konistas el paroj de naturlingvaj frazoj kaj la respektivej tradukoj al la Interchange Format, oni ankaŭ ĉi tie hezonas ekspertojn por la kolektro de la korpuso. Tio motivigas plaujn konfiderojn, celante al lernado de la necesaj parametroj el korpuso kun du naturaj lingvoj. La ideo estas, ke modelo povus kreui korpuson por la Germana kaj Interchange Format el du korpusoj: German-Angla kaj korpuso por la Angla kaj Interchange Format.

La ekkonfio trovatas dum nia laboro reaktigis en programo por ebligi komparon kun kutime uzataj metodoj. Tiel evidentigis, ke – malgraŭ la kompare simpla metodo kaj la nur malgranda trejnkorporado – riceviĝis surprize bonaj rezultoj. Samtempe la rezultoj de la eksperimentoj aperigis novajn demandoj, kiuj indikas interesajn eblecojn por plua esplorado.
# Inhaltsverzeichnis

Zusammenfassung 5

Abstract 7

Résumé 9

Resumo 11

1 Einleitung 15

2 Statistische Methoden 19
   2.1 Das Modell des verrauschten Kanals .................... 19
   2.2 Ein einfaches Sprachmodell ............................ 21
   2.3 Klassische Übersetzungsmodelle ......................... 24
   2.4 Ein Algorithmus zum Decoding .......................... 28

3 Linguistische Methoden 31
   3.1 Allgemeine Überlegungen ............................... 31
   3.2 Das Interchange Format ................................. 32
   3.3 Sprachübersetzung im Nespole! Projekt ................ 35

4 Statistisch ins Interchange Format 37
   4.1 Formale Grundlagen ..................................... 37
   4.2 Ein Sprachmodell für Termsprachen ...................... 38
   4.3 Übersetzungsmodelle für Termsprachen .................. 40
   4.4 Decodierung ............................................. 43
   4.5 Besonderheiten beim Interchange Format ................ 46

5 Projektion von Übersetzungsmodellen 49
   5.1 Projektion von IBM-1-Modellen ......................... 50
   5.2 Projektion von IBM-2-Modellen ......................... 51

6 Das System JTrans 53
   6.1 Grundkonzept des Systems .............................. 53


<table>
<thead>
<tr>
<th>6.2 Basisfunktionalität</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3 Realisierung der Übersetzung ins Interchange Format</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7 Evaluation</th>
<th>59</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Vergleich verschiedener Zuordnungsalgorithmen</td>
<td>59</td>
</tr>
<tr>
<td>7.2 Qualität der erzeugten Terme</td>
<td>62</td>
</tr>
<tr>
<td>7.3 Vergleichende Humanevaluation</td>
<td>66</td>
</tr>
<tr>
<td>7.4 Verluste durch Projektion</td>
<td>69</td>
</tr>
<tr>
<td>7.5 Einfluß der Corpusgröße</td>
<td>71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8 Ergebnisse und offene Fragen</th>
<th>75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zum Nespole Corpus</td>
<td>79</td>
</tr>
<tr>
<td>Evaluationsdaten</td>
<td>83</td>
</tr>
<tr>
<td>Weiteres zu JTrans</td>
<td>91</td>
</tr>
<tr>
<td>Iterationsalgorithmen</td>
<td>101</td>
</tr>
<tr>
<td>Begleitmaterial</td>
<td>105</td>
</tr>
<tr>
<td>Symbolverzeichnis</td>
<td>107</td>
</tr>
<tr>
<td>Literatur</td>
<td>109</td>
</tr>
<tr>
<td>Index</td>
<td>111</td>
</tr>
</tbody>
</table>

Erklärung


Karlsruhe, 31. Mai 2002

Manuel Kauers
Das Problem der automatisierten Sprachübersetzung ist

Den derzeitigen Übersetzungssystemen liegen zwei grundlegend verschiedene Herangehensweisen zugrunde. Die auf Warren Weaver zurückgehende statistische Übersetzung (Hutchings, 1997) versucht, anhand eines bilingualen Corpus Parameter eines statistischen Modells zu lernen, mit deren Hilfe zu jedem Paar \((e,f)\) von Sätzen eine approximierte Wahrscheinlichkeit dafür angegeben wird, daß \(e\) eine gültige Übersetzung des Satzes \(f\) ist. Der Vorgang des Übersetzens besteht dann in der Suche nach einem Satz \(e\), für den die Wahrscheinlichkeit \(p(e|f)\) zu vorgegebenem \(f\) maximal wird (Kapitel 2).

Demgegenüber steht eine Vielzahl verschiedener Verfahren, die in dieser Arbeit unter dem Begriff linguistische Methoden zusammengefaßt werden sollen. Ihnen ist gemeinsam, daß sie den Übersetzungsprozeß als einen zweistufigen Prozess darstellen: Das Verstehen des Gesagten (d.h. die Überführung des semantischen Gehalts eines in natürlicher Sprache vorliegenden Textes in eine sprachunabhängige Repräsentation, in eine Interlingua) und die Generierung eines naturnäheren Textes der Zielsprache aus dieser Repräsentation (Kapitel 3).

Statistische Verfahren zeichnen sich gegenüber linguistischen durch ihre Robustheit und ihre Flexibilität aus. Ein statistisches Übersetzungssystem, das hinreichend allgemein entworfen ist, läßt sich auch auf Sprachpaare anwenden, an die zur Zeit seiner Entwicklung noch nicht gedacht wurde. Eine Bedingung ist, daß ein genügend großes bilinguales Corpus für das gewünschte Sprachpaar zur Verfügung steht.

Abbildung 1 Veranschaulichung der Komplexitätsreduktion beim Einsatz einer Interlingua


Die lineare Form der Konzeptketten ermöglicht es Macherey et al. (2001), ein bestehendes statistisches Übersetzungssystem ohne weitere Veränderungen auf das Sprachpaar Deutsch/Konzeptketten anzuwenden. Die meisten bestehenden Zwischensprachen sind aber nicht linear sondern baumartig strukturiert und lassen sich daher nicht ohne weiteres wie natürliche Sprachen behandeln. Die Hauptaufgabe, der sich die vorliegende Arbeit widmet, ist daher die Weiterentwicklung von etablierten statistischen Sprach- und Übersetzungsmodellen, so daß damit sowohl linear als auch termartig aufgebaute Sprachen behandelt werden können. In der Weise, wie man eine Kette

---

1Zwar haben auch natürlichsprachliche Formulierungen nicht wirklich eine lineare Struktur, doch liegt die Nichtlinearität von natürlicher Sprache nicht an der Oberfläche, so daß sie rein statistischen Verfahren verborgen bleibt.
als speziellen Baum auffassen kann, werden die zu erarbeitenden Erweiterungen Verallgemeinerungen der klassischen Modelle darstellen.


Zwar ist zu vermuten, daß die Annotation einer Trainingsmenge mit Interlingua-Repräsentationen bereits wirtschaftlicher möglich ist als die Entwicklung einer Grammatik für eine natürliche Sprache, doch sind für diese Arbeit nach wie vor Experten nötig, die sowohl die natürliche Sprache als auch die verwendete Interlingua beherrschen. Dies führt auf den Wunsch nach einem Verfahren, mit dem man ein Modell zur Übersetzung einer Sprache $A$ in Interlingua ($A \rightarrow IL$) durch Verkettung eines Modells für $A \rightarrow B$ mit einem bestehenden Modell für $B \rightarrow IL$ erhalten kann. Ein solches Vorgehen wird als Projektion bezeichnet und für die von uns verwendeten Wahrscheinlichkeitsmodelle in Kapitel 5 vorgestellt.

\[ \begin{align*}
\text{Ich würde gerne von Köln nach München fahren} \\
\text{want_question \ origin \ destination \ going}
\end{align*} \]

Abbildung 2 Eine Zuordnung von Wörtern eines deutschen Quellsatzes zu Konzeptbezeichnern, vgl. Macherey et al. (2001), Abbildung 1

Übersicht

Die vorliegende Diplomarbeit ist wie folgt aufgebaut:

Kapitel 2 und 3 tragen die erforderlichen Grundlagen aus der statistischen und linguistischen Übersetzung zusammen.


Kapitel 5 beschäftigt sich im Anschluß daran mit der Projektion eines natürlichsprachen Übersetzungsmoellns auf ein Modell zur Übersetzung nach Interlingua.

Weitere Kapitel der Arbeit beschreiben eine Implementierung (Kapitel 6) und experimentelle Ergebnisse (Kapitel 7). Den Abschluß bildet eine Zusammenfassung der Ergebnisse und ein Ausblick auf weitere möglich Betätigungsfelder (Kapitel 8).
Dank

Ich danke Donna Gates, Kay Peterson und Dorcas Wallace für ihre technische Unterstützung beim Vergleich zum grammatisch basierten Ansatz (Abschnitt 7.3).
Frederik Thiele und Dominique Unruh sei gedankt für ihre Übersetzungen der Zusammenfassung in Französisch (S. 9) bzw. Esperanto (S. 11).
Zu danken ist ferner Benjamin Bertram, Christian Fügen, Christoph Mönch-Tegeder, Jürgen Reichert, Thomas Schaal und Christian Wiedenhoff, die als Humanevaluatoren zur Verfügung standen.
Schließlich danke ich Christian Fügen, Jürgen Reichert, Thomas Schaal und besonders Stephan Vogel und Alex Waibel für ihre inhaltlichen Beiträge und ihre Diskussionsfreudigkeit.
2 Statistische Methoden

2.1 Das Modell des verrauschten Kanals


Das der statistischen Kryptoanalyse entlehnte Modell des verrauschten Kanals basiert auf der Vorstellung, ein aus dem Deutschen ins Englische zu übersetzender Satz sei vom Sprecher ursprünglich bereits auf Englisch (der Zielsprache!) formuliert worden, und kuriose Störungen des Übertragungskanals hätten derart auf den Satz eingewirkt, daß beim Empfänger eine deutsche Version des ursprünglich englischen Satzes ankam (Abb. 3). Die Idee der statistischen Übersetzung ist demnach, durch Training die Störungen des Übertragungskanals zu „begriffen“, um später aus dem gestörten Signal (d.h. aus dem Quellsatz) das ungestörte Ursprungssignal (d.h. den Zielsatz) zu rekonstruieren.

Abbildung 3 Das Modell des verrauschten Kanals: Der Hörer, dessen Sprache (hier Englisch) von der des Sprechers (hier Deutsch) verschieden ist, glaubt, der Sprecher spräche sehr wohl die gleiche Sprache wie er. Daß er den Sprecher dennoch nicht versteht, erklärt er sich durch Störungen des Übertragungskanals.
Bezeichnet man mit \( p(e|f) \) die Wahrscheinlichkeit dafür, daß ein menschlicher Experte den Satz \( e \) als Übersetzung des Satzes \( f \) nennt, dann läßt sich Übersetzung beschreiben als die Suche nach einem Satz \( \hat{e} \), so daß für vorgegebenes \( f \) die Wahrscheinlichkeit \( p(\hat{e}|f) \) maximal wird:

\[
\hat{e} := \arg \max_e p(e|f).
\]

Zur Vereinfachung der Schreibweise wird im folgenden nicht explizit zwischen der „wahren“ Wahrscheinlichkeit \( p \) und ihrer Approximation durch ein mathematisches (statistisches) Modell unterschieden.\(^2\)

Eine elegante Möglichkeit zur Modularisierung ergibt sich durch Anwendung der Regel von Bayes. Nach ihr gilt bekanntlich

\[
p(e|f) = \frac{p(e)}{p(f)} p(f|e),
\]

und da bei der Suche nach \( \hat{e} \) die Wahrscheinlichkeit \( p(f) \) konstant bleibt, erhält man

\[
\hat{e} = \arg \max_e p(e|f) = \arg \max_e (p(e)p(f|e)).
\]

(1)

In dieser Fassung lassen sich den beiden Faktoren auf der rechten Seite intuitiv zwei unabhängige „Aufgaben“ zuordnen: \( p(e) \) modelliert die Wahrscheinlichkeit, daß eine Wortfolge \( e \) tatsächlich auch ein wohlgeformter Satz ist. Man erwartet etwa

\[
p(\text{good morning , sir .}) > p(\text{morning , sir . good}).
\]

Ein statistisches Modell zur Approximation von \( p(e) \) wird entsprechend als Sprachmodell bezeichnet (vgl. Abschnitt 2.2). Den zweiten Faktor, \( p(f|e) \), könnte man intuitiv als den semantischen Abstand zwischen den Wortfolgen \( f \) und \( e \) deuten. Man erwartet also hier etwa

\[
p(\text{Guten Morgen .} \mid \text{good morning , sir .})
\]

\[
> p(\text{Auf Wiedersehen .} \mid \text{good morning , sir .})
\]

und spricht bei einer Modellierung dieser Wahrscheinlichkeit vom Übersetzungsmodell.\(^3\)

Es bleibt also nunmehr das Problem, allgemeine Sprach- und Übersetzungsmodelle zu formulieren, deren Parameter sich durch Analyse eines bilingualen Corpus algorithmisch schätzen lassen (Training). Damit beschäftigen sich die folgenden beiden Abschnitte. Ein weiteres Problem beschränkt den Suchraum: Zu jedem Quellsatz \( f \) gibt es unendlich viele – bei Beschränkung der Satzlänge immer noch exponentiell viele – potentielle Übersetzungen \( e \), unter denen die optimale Übersetzung \( \hat{e} \) auszuwählen ist. Da eine volle Suche ausscheidet, sind spezielle Suchalgorithmen nötig. Ein solcher Algorithmus zum sogenannten Decoding wird in Abschnitt 2.4 vorgestellt.

---

\(^2\) Im übrigen orientiert sich die Nomenklatur in dieser Arbeit an den Bezeichnungsweisen, die sich seit Brown et al. (1993) in der Literatur weitgehend eingebürgert haben. Im Zweifelsfall sei auf das Symbolverzeichnis auf Seite 107 verwiesen.

\(^3\) Hier werden wie in der statistischen Übersetzung üblich Interpunktionzeichene wie einzelne Wörter behandelt.
2.2 Ein einfaches Sprachmodell


Die Grundlage der meisten Sprachmodelle ist die Zerlegung

\[ p(e) = \prod_{i=1}^{l} p(e_i | e_1, \ldots, e_{i-1}), \]

(2)

nach der die Wahrscheinlichkeit für eine Sequenz \( e = (e_1, \ldots, e_l) \) dargestellt wird als Produkt der Wahrscheinlichkeiten für jedes der \( e_i \) unter der Voraussetzung, daß zuvor die Sequenz \( e_1, \ldots, e_{i-1} \) gesehen wurde. Gleichung (2) umschreibt mathematisch, was in der Linguistik unter der Bezeichnung Voraktivierung bekannt ist: Aufgrund des bisher Gelesenen oder Gehörten baut der menschliche Sprachersteller eine Erwartungshaltung darauf auf, welche Wörter wohl als nächstes erscheinen werden. Die These von der Voraktivierung während der Sprachaufnahme beim Menschen läßt sich durch verschiedene Alltagsbeobachtungen stützen. So erlaubt sie zum Beispiel einem aufmerksamen Zuhörer, dem ins Stocken geratenen Sprecher das Wort zu nennen, das ihm gerade fehlt. Ferner trägt sie zur Erleichterung des Leseprozesses bei: Statt daß ein Wort exakt aus seiner Buchstabenfolge konstruiert wird, wird das Schriftbild des Gesamtwortes abgeglichen mit den voraktivierten Wörtern. Ähnlichkeit genügt dabei schon, um ein Wort als „erkannt“ anzusehen, was es unter anderem so schwer gemacht, Tippfehler in einem Text zu finden. Erscheint ein Wort, das nicht voraktiviert war, so wird dies vom Mensch spontan als witzig empfunden. (Zur Voraktivierung vgl. auch Linke et al., 1996)

Da die Maschine so wenig wie der Mensch in der Lage ist, beliebig lange Historien zu berücksichtigen, und weil die Annahme plausibel erscheint, daß weit zurückliegende Wörter einen geringeren Einfluß auf die Wahl der folgenden Wörter haben, beschränkt man sich bei der Modellierung auf sogenannte \( n \)-Gramme. Die Wahrscheinlichkeiten \( p(e_i | e_1, \ldots, e_{i-1}) \) werden dabei approximiert durch die relativen Häufigkeiten

\[ p(e_i | e_1, \ldots, e_{i-1}) \approx \frac{n(e_i | e_{i-n}, \ldots, e_{i-1})}{\sum_{\varepsilon} n(e_{i-n}, \ldots, e_{i-1}, \varepsilon)} \]

aller im Trainingsmaterial gesehenen \( n \)-Gramme \( (e_{i-n}, \ldots, e_i) \). Steht hinreichend viel Trainingsmaterial zur Verfügung, so steigt mit wachsendem \( n \) (nach dem Speicherbedarf) auch die Präzision der Schätzung. Andererseits können zu groß gewählte \( n \) zusammen mit zu kleinen Trainingsdatensätzen durch übermäßiges Rauschen die Qua-
lität der Schätzung wieder mindern. Typische Werte für n sind 2 (Bigramm) oder 3 (Trigramm).

Problematisch wird die Verwendung von relativen Häufigkeiten, wenn die Wahrscheinlichkeit eines n-Gramms geschätzt werden soll, das in den Trainingsdaten gar nicht vorgekommen ist. Nähe man der relativen Häufigkeit folgend für diese Ereignisse eine Wahrscheinlichkeit von 0 an, so würde diese das gesamte Produkt zu 0 machen und damit die möglicherweise bessere Schätzung der Umgebung vernichten. Die folgende Rechnung macht dabei deutlich, daß z. B. ungesuchene Trigramme keine Seltenheit sind. Grob vereinfachend nehme man dazu an, daß alle der m in einer Sprache möglichen Trigramme gleich wahrscheinlich sind, so daß die Analyse eines Trainingscorpus durch Ziehen mit Zurücklegen modelliert werden kann. Gesucht ist die Wahrscheinlichkeit \( p \), nach \( k \)-maligem Ziehen (Corpusgröße \( k \)) alle Kugeln (Trigramme) der Urne (Sprache) einmal gezogen zu haben. Über das Gegenereignis, mindestens eine der Kugeln nach \( k \) Zügen immer noch nicht gesehen zu haben, erhält man

\[
p = 1 - \left( \frac{m - 1}{m} \right)^k.
\]

Damit diese Wahrscheinlichkeit über eine vorgegebene Schranke \( \alpha \) steigt, sind folglich

\[
k \geq \frac{\log(1 - \alpha)}{\log(m - 1) - \log m}
\]

Ziehungen nötig. Legt man nun eine realistische Lexikongröße von \( 10^7 \) Wörtern zugrunde, so ergeben sich \( m = 10^{15} \) Trigramme. Nach obiger Rechnung wäre also ein Trainingscorpus von ca. \( 2,31 \cdot 10^{15} \) Wörtern nötig, wenn die Wahrscheinlichkeit, jedes Trigramm einmal gesehen zu haben, über 90% steigen soll.\(^1\) Konkrete Untersuchungen am Nespolo!- Corpus (vgl. S. 79), die zeigen, daß im Englischen bis zu 40% der Trigramme aus den Testmengen nicht in der Trainingsmenge vorhanden waren, sind danach nicht mehr überraschend.

Um das Problem mit den Nullwahrscheinlichkeiten zu umgehen, wendet man einen Kunstgriff an, der unter der Bezeichnung Backing Off bekannt ist. Dabei wird ein gewisser Teil \( \alpha \) der Wahrscheinlichkeitsmasse für ungesuchte n-Gramme zurückgehalten. Man erhält dann

\[
p'(e_i | e_{i-n}, \ldots, e_{i-1}) = \begin{cases} (1 - \alpha) p(e_i | e_{i-n}, \ldots, e_{i-1}) & \text{falls } n(e_{i-n}, \ldots, e_{i-1}) \geq n_0 \\ \alpha & \text{sonst} \end{cases}
\]

Formal betrachtet man alle ungesuchten (bzw. allgemeiner: selten gesehenen) Tupel als gleich und ordnet diesen die zurückgehaltene Wahrscheinlichkeit \( \alpha \) zu.

Als weitere Verbesserung bietet es sich an, auch die relativen Häufigkeiten von \( (n-1) \)-Grammen, \( (n-2) \)-Grammen usw. in die Schätzung eingehen zu lassen. Dies geschieht üblicherweise durch Interpolation mit dem Schätzwert für kleinere \( n \):

\[
p''(e_i | e_{i-n}, \ldots, e_{i-1}) = \sum_{k=0}^{n} \lambda_k p'(e_i | e_{i-k}, \ldots, e_{i-1})
\]

\( ^1\) Bei einer optimalen Codierung von 7 Bytes je Trigramm wären zur Speicherung eines derartig großen Corpus 45 159 284 CDs à 680 MB nötig.
für geeignet gewählte Interpolationsgewichte $\lambda_0, \ldots, \lambda_n \in [0, 1]$ mit $\sum \lambda_j = 1$.

Zur Bewertung der Qualität eines Sprachmodells bedient man sich eines Konzepts aus der Codierungslehre. Die \textit{Perplexität} $P(T)$ einer Menge $T$ von $n$-Grammen ist definiert durch

$$P(T) := \frac{1}{\sqrt[n]{\prod_{t \in T} p(t)}} = e^{-\frac{1}{|T|} \sum_{t \in T} \log p(t)}.$$ 


Die Perplexität eröffnet nun eine einfache Möglichkeit zur Schätzung von Interpolationsparametern. Die Trainingsmenge $T$ wird in zwei Teile $A$ und $B$ aufgeteilt. Die (typischerweise größere) Menge $A$ dient zunächst zur Bestimmung der relativen Häufigkeiten. Danach wählt man die Interpolationsgewichte derart, daß die Perplexität der ungesuchten Menge $B$ minimiert wird. Dazu bietet sich eine rekursive Formulierung der Interpolation an:

$$p''(e_i) := p'(e_i),$$

$$p''(e_i | e_{i-n}, \ldots, e_{i-1}) := \lambda_0 p'(e_i | e_{i-n}, \ldots, e_{i-1}) + (1 - \lambda_0) p'(e_i | e_{i-n+1}, \ldots, e_{i-1}).$$

Diese Gleichungen über die sich $\lambda_1$ bis $\lambda_n$ nacheinander bestimmen lassen. Die zunächst provisorisch bestimmten relativen Häufigkeiten können nach Schätzung der Interpolationsparameter durch die entsprechenden Werte von ganz $T$ ersetzt werden.

In der Sprachmodellimplementierung dieser Diplomarbeit (vgl. Kapitel 6) wird das Trainingscorpus $T$ in $m$ gleich große Partitionen $T_1, T_2, \ldots, T_m$ aufgeteilt, aus denen sich $m$ Zerlegungen

$$A_i = \bigcup_{j=1}^{m} T_j, \quad B_i = T_i \quad (i = 1, \ldots, m)$$

ergeben. Für jede dieser Zerlegungen werden unabhängig voneinander Interpolationsparameter $\lambda^{(i)}_0, \ldots, \lambda^{(i)}_n$ ermittelt, deren arithmetisches Mittel zu den schließlich verwendeten Werten führt:

$$\bar{\lambda}_k := \frac{1}{m} \sum_{i=1}^{m} \lambda^{(i)}_k \quad (k = 1, \ldots, n).$$

Im Falle des speziellen $m = |T|$, wenn also jedes $T_i$ nur ein Element beinhaltet, spricht man vom \textit{Leaving One Out}.

2.3 Klassische Übersetzungsmodelle

Brown et al. (1993) stellen fünf aufeinander aufbauende Übersetzungsmodelle vor, die unter der Bezeichnung IBM 1 bis IBM 5 bekannt sind, und von denen in diesem Abschnitt die Modelle IBM 1 und IBM 2 vorgestellt werden sollen. Zentrales Konzept dieser Modelle ist die Zuordnung (engl. Alignment), bei der man annimmt, daß sich in Sätzen \( (e,f) \), die Übersetzungen voneinander sind, die meisten Wörter oder Wortgruppen des einen Satzes bestimmten Wörtern oder Wortgruppen des anderen Satzes zuordnen lassen. Bereits in Abbildung 2 ist ein Beispiel für eine Zuordnung gezeigt, Abbildung 4 zeigt ein Beispiel für zwei natürlichsprachliche Sätze.

Und was kostet das Ganze ?

| And | how much does that cost ? |

Abbildung 4 Beispielhafte Zuordnung zwischen den Wörtern zweier entsprechender Sätze des Deutschen und Englischen

In ihrer allgemeinsten Form kann man Zuordnungen formal nur als Relationen von Tokenindizes auf Quell- und Zielseite fassen, doch auch durch Verwendung des bequemeren Funktionsbegriffs lassen sich für die meisten Satzpaare bereits plausible Zuordnungen formulieren. Man schreibt eine Zuordnung zwischen einem Satz \( f \) der Länge \( m \) und einem anderen Satz \( e \) der Länge \( l \) als Funktion \( a: \{1, \ldots, m\} \rightarrow \{0, \ldots, l\} \) und sagt, die Positionen \( i \) und \( j \) seien einander zugeordnet, falls \( a_j := a(j) = i \) gilt. Im Fall \( a_j = 0 \) sei der \( j \)-ten Position kein Wort der anderen Seite zugeordnet. (Jeder Satz \( e = (e_1, \ldots, e_l) \) wird dazu formal um ein leeres Wort an Position \( i = 0 \) ergänzt.)

Die Übersetzungsmodelle IBM 1 und IBM 2 zur Schätzung von \( p(f|e) \) beruhen auf der Zerlegung

\[
p(f|e) = \sum_a p(f, a|e)
\]

mit

\[
p(f, a|e) = p(m|e) \prod_{j=1}^{m} \left( p(a_j|a_{j-1}^{j-1}, f_j^{j-1}, m, e) p(f_j|a_{j-1}^{j-1}, m, e) \right).
\]

Dabei steht \( x_j \) kurz für \( x_{e_j}, \ldots, x_{e_j} \). Diese Zerlegung fußt auf der Vorstellung, daß man, \( f \) aus \( e \) generierend, zunächst die Länge \( m \) von \( f \) bestimmt, um dann für jedes \( j = 1, \ldots, m \) zunächst die zugeordnete Zielseite \( a_j \) und dann das Quellwort \( f_j \) an der Position \( j \) als Übersetzung des Zielwortes \( e_{a_j} \) zu bestimmen.\(^5\)

Brown et al. (1993) betrachten Gleichung (3) als exaktes Modell, dessen verschieden starke Vereinfachungen zu den verschiedenen IBM-Modellen führen (,,It is important to realize that Equation (3) is not an approximation."). In den Modellen wird die Zahl der zu trainierenden Parameter dann soweit reduziert, daß sie sich mit Corpora realistischer Größe hinreichend gut schätzen lassen. So verwendet das Modell IBM 1 nur

\(^5\) Man beachte, daß diese Motivation nicht als Grundlage eines Decoders geeignet ist, da hier \( f \) aus \( e \) generiert wird, während wir insgesamt an der umgekehrten Richtung interessiert sind.
Übersetzungswahrscheinlichkeiten (engl. translation probabilities) \( t(f|e) \):

\[
p(f|e) := \sum_a \frac{1}{(l+1)^m} \prod_{j=1}^m t(f_j|e_a_j)
\]

\[
= \frac{1}{(l+1)^m} \sum_{a_1=0}^l \sum_{a_2=0}^l \ldots \sum_{a_m=0}^l \prod_{j=1}^m t(f_j|e_{a_j})
\]

\[
(Brown\ et\ al.,\ 1993) \quad \frac{1}{(l+1)^m} \prod_{j=1}^m \sum_{e_j} t(f_j|e_j)
\]

Intuitiv steht \( t(f|e) \) etwa für die Wahrscheinlichkeit, dass ein Wort \( f \) zu einem Wort \( e \) zugeordnet wird, also gewissermaßen dessen direkte Übersetzung ist.

Die Schätzung der Parameter \( t(f|e) \) läßt sich auffassen als Suche nach der Funktion \( t \), die \( p(f|e) \) für vorgegebene \( (f, e) \) unter der Nebenbedingung

\[
\sum_f t(f|e) = 1
\]

maximiert. Nach dem Satz von Lagrange verschwindet für eine solche Funktion \( t \) die partielle Ableitung der Hilfsfunktion

\[
h(t(f|e), \lambda) = p(f|e) - \sum_e \lambda_e \left( \sum_f t'(f|e') - 1 \right)
\]

nach \( t(f|e) \). Schreibt man die Ableitungsregel für ein Produkt von Funktionen in der Form

\[
\left( \prod_j g_j \right)' = \left( \prod_j g_j \right) \cdot \sum_j \frac{g_j'}{g_j}
\]

(unter der Annahme \( g_j(x) \neq 0 \)) und verwendet die Darstellung (4.c) von \( p(f|e) \), so erhält man

\[
\frac{\partial}{\partial t(f|e)} h(t(f|e), \lambda) = p(f|e) \cdot \sum_{j=1}^m \frac{\delta(f, f_j) \delta(e, e_j)}{\sum_{j=0}^l t(f_j|e_j)} - \lambda_e = 0
\]

\[\iff\]

\[
t(f|e) = \lambda_e^{-1} p(f|e) \cdot \frac{t(f|e)}{\sum_{j=0}^l t(f_j|e_j)} \sum_{j=0}^l \delta(f, f_j) \sum_{i=0}^l \delta(e, e_i).
\]

(5)

Die durch

\[
c(f|e, f, e) := \frac{t(f|e)}{\sum_{i=0}^l t(f_i|e_i)} \sum_{j=1}^l \delta(f, f_j) \sum_{i=0}^l \delta(e, e_i) = \lambda_e^{-1} \frac{t(f|e)}{p(f|e)}
\]

definierten Counts lassen sich interpretieren als Erwartungswert für die Anzahl der Verbindungen zwischen Wörtern \( f \) und \( e \) im Satzpaar \( (f, e) \). Offenbar erhält man also
die \( t(f|e) \) durch Normierung dieses Wertes. Die Linearität des Erwartungswerts legitimiert für eine Menge \( C \) von Satzpaaren die Trainingsgleichung

\[
t(f|e) = \lambda'_c \sum_{(f,e) \in C} c(f|e, f, e)
\]

für geeignete Normierungskonstanten \( \lambda'_c \).

Da die Counts \( c(f|e, f, e) \), aus denen nach der obigen Trainingsgleichung die \( t(f|e) \) gewonnen werden sollen, ihrerseits selbst von den \( t(f|e) \) abhängen, ist ein iteratives Training nötig (Algorithmus 1). Dieses konvergiert, wie Brown et al. (1993) zeigen, gegen ein eindeutiges Optimum, so daß die Initialisierung der \( t(f|e) \) irrelevant ist.

```
1  function ibm1Training(Corpus, number of iterations)
2      begin
3         // let L be the set of all possible words (source lexicon)
4         for all e, f do // initialize t arbitrarily
5             t(f|e) := 1/L.size();
6         for n = 1 to number of iterations do
7             for all e, f do // initialize counts to zero
8                 c(e, f) := 0;
9             for (e, f) \in Corpus do // determine counts
10                for j = 1 to m do
11                    sum := \sum_{i=0}^j t(f_j|e_i);
12                    for i = 0 to j do
13                        c(e_i, f_j) := c(e_i, f_j) + t(f_j|e_i)/sum;
14                for all e, f \in L do // redefine translation probs as normalized counts.
15                   t(f|e) := \lambda'_c c(e, f);
16         return t;
17     end
```

Algorithmus 1 Algorithmus zum Training der Parameter des IBM 1 Übersetzungsmodells

Das Modell IBM 2 verwendet neben den Übersetzungswoahrscheinlichkeiten positions-abhängige Zuordnungswoahrscheinlichkeiten (engl. alignment probabilities). Durch die zusätzlichen Parameter \( a(i|j, m, l) \) wird intuitiv die Wahrscheinlichkeit formuliert, daß \( a_j = i \) ist, also das \( j \)-te Wort in \( f \) dem \( i \)-ten Wort in \( e \) zuzuordnen ist. Damit ergibt sich

\[
p(f|e) := \sum_a \prod_{j=1}^m t(f_j|e_{a_j})a(a_j|j, m, l)
\]

(6a)

\[
= \sum_{a_1=0}^l \sum_{a_2=0}^l \ldots \sum_{a_m=0}^l \prod_{j=1}^m t(f_j|e_{a_j})a(a_j|j, m, l)
\]

(6b)

(\text{Brown et al., 1993})

\[
\sum_{j=1}^m \prod_{j=1}^l t(f_j|e_j)a(i|j, m, l)
\]

(6c)
Im Training sind nun neben $r(f|e)$ auch die Parameter $a(i,j,m,l)$ zu maximieren. Dazu wendet man wieder den Satz von Lagrange an, diesmal auf die Hilfsfunktion

$$h(t,a,\lambda,\mu) = p(f|e) - \sum_{e'} \lambda_{e'} \left( \sum_f r(f'|e') - 1 \right) - \sum_f \mu_{jml} \left( \sum_{e_f} a(f'|j,m,l) - 1 \right).$$

Dies führt mit (6,6) auf

$$\frac{\partial}{\partial t(f|e)} h(t,a,\lambda,\mu) = p(f|e) \cdot \sum_{j=1}^m \sum_{e_i} r(f|e_i) a(i,j,m,l) \delta(f,f_j) \delta(e,e_i) - \lambda_e = 0$$

$$i(f|e) = \lambda_e^{-1} p(f|e) \sum_{j=1}^m \sum_{e_i} r(f|e_i) a(i,j,m,l) \delta(f,f_j) \delta(e,e_i) \sum_{i=0}^l \sum_{e_i} r(f|e_i) a(i,j,m,l)$$

bzw. für die Zuordnungswahrscheinlichkeiten auf

$$\frac{\partial}{\partial a(i,j,m,l)} h(t,a,\lambda,\mu) = p(f|e) \sum_{j=1}^m \sum_{k=0}^l r(f_j|e_k) a(k,j,m,l) - \mu_{jml}$$

$$a(i,j,m,l) = \mu_{jml}^{-1} p(f|e) \sum_{j=1}^m \sum_{k=0}^l r(f_j|e_k) a(i,j,m,l) \sum_{i=0}^l \sum_{e_i} r(f|e_i) a(i,j,m,l)$$

Daraus ergeben sich wie oben die Counts

$$c(f|e,f,e) = \sum_{j=1}^m \sum_{i=0}^l \sum_{e_i} r(f|e_i) a(i,j,m,l) \delta(f,f_j) \delta(e,e_i)$$

$$c(i,j,m,l,f,e) = \frac{r(f_j|e_l) a(i,j,m,l)}{\sum_{i=0}^l \sum_{e_i} r(f_j|e_i) a(k,j,m,l)}$$

und damit die Trainingsgleichungen

$$r(f|e) = \lambda_e \sum_{(t,e) \in C} c(f|e,f,e), \quad a(i,j,m,l) = \mu_{jml} \sum_{(t,e) \in C} c(i,j,m,l,f,e)$$

für geeignet gewählte Normierungsconstanten $\lambda_e$ und $\mu_{jml}$. Algorithmus 1 ändert sich entsprechend. Da sich das IBM-1-Modell als spezielles IBM-2-Modell auffassen läßt, in dem $a(i,j,m,l) = 1/(l+1)$ gilt, lassen sich zur Initialisierung des IBM-2-Trainings die Ergebnisse eines zuvor trainierten IBM-1-Modells verwenden.
Im Unterschied zu den Modellen 1 und 2 wird in höheren Modellen ein mächtigerer Zuordnungsbegriff verwendet. Es wird nunmehr festgestellt, daß mehrere Wörter des Quellsatzes mehreren Wörtern des Zielsatzes zugeordnet werden. Der entscheidende Nachteil dabei ist, daß zum Schätzen der Parameter und zum späteren Decodieren im Gegensatz zu den ersten beiden Modellen keine effizienten Algorithmen bekannt sind.

In Modell 3 wird zu jedem Quellwort $e_i$ eine zusätzliche Wahrscheinlichkeit $n(q_i|e_i)$ dafür eingeführt, daß dieses Wort $q_i$ Wörter im Zielsatz entsprechen. Man spricht von *Fertilities*. Die Modelle 4 und 5 sind Verfeinerungen von Modell 3, in denen die Position $j$, der das Quellwort $e_i$ zugeordnet wird, zusätzlich von den *Wortklassen* abhängt, in denen $f_j$ und $e_i$ liegen.


Ähnlich wie beim Sprachmodell (Abschnitt 2.2) lassen sich auch die für das Übersetzungsmodell geschätzten Parameter durch Interpolation und Backing Off leicht verbessern. So erreicht man durch Interpolation der Übersetzungserscheinlichkeiten $t$ mit einer Gleichverteilung, mit $L_F$ als Lexikon der Quellsprache also

$$t'(f|e) := \lambda t(f|e) + (1 - \lambda) \frac{1}{|L_F|},$$

und/oder durch Einsatz von Backing-Off-Mechanismen eine größere Robustheit gegenüber Wörtern, die im Trainingscorpus nicht oder selten gesehen wurden. Analog zum Vorgehen dort läßt sich auch eine Perplexität des Übersetzungsmodells definieren. Sie drückt aus, wie groß im Mittel der Pool an Quellsprachwörtern pro Zielsprachwort wäre, bei dem eine zufällige Wahl des übersetzten Wortes mit gleicher Wahrscheinlichkeit zum richtigen Ergebnis führen würde wie die Vorhersage des Übersetzungsmodells. Für ein IBM-2-Modell führt dies auf die Definition

$$P(T) := \exp \left( -\frac{1}{|T|} \sum_{(f,e) \in T} \log \sum_{i=0}^{m} t(f_j|e_i) \alpha(i|j,m,t) \right).$$

Modellunabhängig läßt sich zusätzliche die *Phrasenperplexität* definieren. Für sie werden Sätze als atomar betrachtet, so daß die Zahlwerte entsprechend größer werden.

$$PP(T) := \exp \left( -\frac{1}{|T|} \sum_{(f,e) \in T} \log p(f|e) \right).$$

### 2.4 Ein Algorithmus zum Decoding

ner Grundlage wird später (Abschnitt 4.4 auf Seite 43) ein Decoder für Baumsprachen entwickelt.

Griindidee des Algorithmus von Wang wie auch vieler ähnlicher Verfahren ist die Verwaltung einer Menge von Hypothesen (Präfixe möglicher Übersetzungen), aus denen durch Hinzufügen neuer Wörter weitere Hypothesen gewonnen werden (Abb. 5). Um eine Explosion des Suchraums zu vermeiden, werden die Hypothesen nach jeder Erweiterung bewertet. Für die nächste Erweiterung behält man dann nur noch die Hypothesen, deren Bewertung eine vorgegebene Schranke \( \alpha \) übersteigt bzw. die \( n \) am besten bewerteten Hypothesen (\( \alpha \) und \( n \) geeignet gewählt).\(^6\)

\[ \sum_e p(e) t(f_j | e) \]

\( \tau(i, j, H) := \begin{cases} t(f_j | e_i) & \text{falls } 0 \leq i \leq k \\ \sum_e p(e) t(f_j | e) & \text{falls } i > k \end{cases} \)


Abbildung 5 Schematische Darstellung der Arbeitsweise des Wang-Decoders
erhält man also
\[
\tau(H) = p(e_1, \ldots, e_k) \cdot \sum_{i=k}^{\infty} p(l|m) \prod_{j=1}^{m} \sum_{i=0}^{j} a(i, j, m, l) \tau(i, j, H)
\]  
(7)
als Bewertung für eine Hypothese \( H = e_1, \ldots, e_k \). Mit \( p(e_1, \ldots, e_k) \) ist dabei die Wahrscheinlichkeit gemeint, die das Sprachmodell (vgl. Abschnitt 2.2) für die Hypothese \( H \) angibt.

Man beachte, daß für die Bewertungsfunktion in (7) die Länge des zu findenden Satzes nicht a priori bestimmt werden muß. Das Satzende wird dadurch erkannt, daß ein spezielles Steuerwort end-of-sentence erscheint, das im Training jedem Satz der Zielsprache angefügt wurde.

Zusammenfassend ergibt sich der folgende Algorithmus:

```
function wangDecoding(f)
    H := \{\lambda\};  // set of hypotheses to be extended further
    R := \emptyset;    // set of complete results found so far
    for k = 1 to maximum target sentence length do
        // generate new hypotheses
        Hnew := \emptyset;
        for all h ∈ H do  // for all hypotheses
            for all e ∈ Lf do  // for all target words
                Hnew := Hnew ∪ h.add(e);
        // cut off less probable hypotheses
        H := \{h ∈ Hnew : τ(H) > α\};
        // generate complete results
        for all h ∈ H do
            R := R ∪ h.add(end of sentence);
        // return the most probable translation
        return arg max_{e∈R} p(e|f);
```

Algorithmus 2 Decoding nach Ye-Yi Wang. Dabei sei \( L_f \) das Lexikon der Zielsprache und \( \lambda \) die leere Hypothese.
3 Linguistische Methoden

3.1 Allgemeine Überlegungen

Die Vorgehensweise eines statistischen Übersetzers unterscheidet sich erheblich von der eines menschlichen Experten. Die Vorstellung, ein Dolmetscher würde zur Übersetzung eines Satzes alle Sätze der Zielsprache durchlaufen, um nach dem Satz zu suchen, der am ehesten mit dem Gesagten zusammenpasst, erscheint absurd. Vielmehr wird ein menschlicher Übersetzer im Normfall versuchen, das Gesagte zunächst zu verstehen, um es dann in der Zielsprache neu zu formulieren. Wenn man also ein System bauen will, das Sprachübersetzung nach menschlichem Vorbild betreibt, muß man zunächst das Verstehen verstehen: Was geschah, wenn der Mensch einen Text versteht?


Zunächst wird aus der (akustisch oder visuell) aufgenommenen Information eine Textrepräsentation in Form einer propositionalen Struktur aufgebaut. Dabei handelt es sich um eine Form der semantischen Darstellung, die der Sprache noch sehr nahe kommt. In ihr sind Präspositionen (implizit vorausgesetzte Bedeutungskomponenten), Deiktika/Pronomen und dergleichen bereits weitgehend aufgelöst, so daß die propositionalen Struktur als rechtfertig anzusehen ist als der ursprüngliche Text.

Der zweite Schritt des Verstehensprozesses ist die Interpretation der propositionalen Struktur, die wegen ihrer größeren Explizitheit nicht vollständig über eine längere Zeit im Gedächtnis behalten werden kann. Dabei wird aus der präpositionalen Struktur ein mentales Modell aufgebaut, in dem die Aussagen des Textes abgebildet werden. Mit fortschreitender Lektüre wird dieses Modell erweitert und nötigenfalls modifiziert, bis schließlich die wesentliche Inhalt des Textes vollständig repräsentiert ist.

Der Aufbau von Textrepräsentation und mentalem Modell ist als eng zueinander verbunden zu verstehen. So können bestimmte Teile des mentalen Modells beim Aufbau der propositionalen Struktur helfen (z. B. bei der Auflösung von Pronomen). Auch können Teile der Textrepräsentation ohne weitere Verarbeitung im Gedächtnis behalten werden, um sie für spätere Interpretationsversuche zu bewahren, wenn sich ihre Aussagen mit dem mentalen Modell in Einklang bringen lassen. Ähnlich werden Textpassagen zum Teil im genauest möglichen Wortlaut abgelegt, wenn sich aus ihnen keine sinnvolle Textrepräsentation erzeugen läßt. Letzteres Phänomen läßt sich durch den experimentellen Befund belegen, daß Versuchspersonen den genauen Wortlaut eines Textes umso leichter vergessen, je besser sie seinen Inhalt verstanden haben. (Zum mentalen Modell im besonderen und zur Vorstellung der Linguistik vom Sprachver-


3.2 Das Interchange Format


\[ \text{speaker} : \text{speech act} + \text{con}_1 + \text{con}_2 + \ldots + \text{con}_n (\text{arg}_1 = \text{val}_1, \ldots, \text{arg}_m = \text{val}_m). \]

Sprecher, Speech Act und die Konzepte bilden zusammen den *Dialog Act*.

Jeder IF-Ausdruck hat genau einen Sprecher. Die beiden möglichen Sprecher sind *a* (für Agent) und *c* (für Customer oder Client).

Die Aktion beschreibt in sehr allgemeiner Weise die Absicht, die der Sprecher mit dem Gesprochenen insgesamt verfolgt, etwa grüßen oder informieren. Ein Aktions-
bezeichner ist für jeden IF-Ausdruck obligatorisch. Er wird gewählt aus einer Liste von aktuell etwa sechzig atomaren Bezeichnern, kann aber durch Voranstellen von verify-, request-verification- und/oder negate- modifiziert werden, so daß insgesamt fast 500 Kombinationen denkbar sind. Kombinierte Speech Acts sind jedoch selten, und auch von den atomaren Speech Acts werden wenige besonders häufig und vie-
le eher selten verwendet. Im Nespoli!-Corpus (vgl. S. 79) sind 97% aller Speech Acts atomar, 71,5% der Ausdrücke haben give-information, acknowledge, request-
information oder affirm als Speech Act.

Dem Speech Act können mehrere Konzeptbezeichner folgen. Durch sie sollen die we-
sentlichen Teile des semantischen Gehalts (etwa Aktionen, Objekte, Attribute, Ereignisse usw.) abgedeckt werden, von denen die Rede ist. Die ca. 110 möglichen Kon-
zeptbezeichner sind nicht beliebig kombinierbar, sie müssen zum Speech Act bzw.
zum vorangegenden Konzept passen. Per Spezifikation lizenstieren Speech Acts und Konzepte, welche Konzepte ihnen folgen dürften. Zur formalen Unterscheidung von Konzept- und Aktionsbezeichnern beginnen Konzeptbezeichner mit dem Symbol „+“. Im Nespoli!-Corpus haben die IF-Ausdrücke zwischen 0 und 5 Konzepte, der Durchschnitt liegt bei 0,9 Konzepten pro Ausdruck. Die am häufigsten auftretenden Konzepte sind +concept, +object, +feature, +existence, +disposition, +feasibility, +action
und +trip.

Argumente und deren Werte codieren schließlich die genaue Information. Argument-
Werte ist der Rahmen (engl. Frame), eine Sammlung von atomaren Werten und Unter-
argumenten, zum Beispiel

\[
\text{room-spec } = (\text{room}, \text{quantity } = 2, \text{for-whom } = i).
\]


Der Unterschied zwischen Relations- und Attributargumenten ist rein semantisch und zur Zeit ohne funktionale Bedeutung. Die Relationsargumente (z. B. for-whom= im obigen Beispiel) stellen Beziehungen zu anderen semantischen Objekten her, während die Attributargumente (z. B. quantity= im obigen Beispiel) nähere Beschreibungen zum Kopf des Rahmens liefern. In Fällen wie

\[
\text{time } = (\text{dow } = \text{friday}, \text{md } = 31, \text{year } = 2002, \text{month } = 5).
\]
ohne einen expliziten Kopf wird implizit ein leerer Kopf angenommen, der von den Attributargumenten (hier dow=, md=, year= und month=) beschrieben wird. Weil
als Köpfe zumeist atomare Werte dienen, und weil die englische Spezifikation den Begriff Value aus historischen Gründen an einigen Stellen sowohl für Werte als auch
für Köpfe verwendet, sei der Unterschied nochmal hervorgehoben: Eine Konstruktion
der Form

\[ \text{quantity} = (5, \text{percentage} = 20) \]

ist nicht gültig, weil die Definition des Arguments \text{quantity}= sowohl die natürlichen
Zahlen als auch das Unterargument \text{percentage}= als Kopf nennt.

Die zweite Art komplexer Werte sind \textit{Listen} (engl. \textit{sets}) von Werten, mit denen alle Ar-
ten von Aufzählungen ausgedrückt werden können. In ihnen können sowohl einfache als auch komplexe Werte auftreten, z. B.

\[ \text{numeral} = [1, 2, 3, 4, 5] \]

für die Aufzählung \textit{1, 2, 3, 4, 5}, oder

\[ \text{room-spec} = [(\text{single}, \text{quantity} = 2), (\text{double}, \text{quantity} = 1)] \]

für \textit{zwei Einzelzimmer, ein Doppelzimmer}. Zumeist bilden Listen zusammen mit dem Argument \text{operator}=, das die Art der Liste spezifiziert, einen Rahmen, etwa

\[ \text{room-spec} = (\text{opera}r = \text{disjunct}, [(\text{single}, \text{quantity} = 2), (\text{double}, \text{quantity} = 1)]) \]

für \textit{zwei Einzelzimmer oder ein Doppelzimmer}. Dabei muß jedes Element der Liste
ein gültiger Wert zum übergeordneten Argument sein. Im Beispiel müssen also neben
\text{operator}= auch single, quantity= und double von \text{room-spec}= lizensiert sein.

Anders als in Rahmen, bei denen die Reihenfolge, in der Werte und Unterargumen-
te angegeben sind, bedeutungslos ist, so daß also \textit{a = (b = c, d = e)} semantisch äquivalent zu \textit{a = (d = e, b = c)} ist, ist die Reihenfolge der Elemente in einer Liste im allgemeinen
deutungsträchtig.

Die Spezifikation stellt insgesamt etwa 300 Argumentbezeichner zur Verfügung, zu
 denen am häufigsten auftretenden gehören identifiable= (zur Unterscheidung von be-
stimmten und unbestimmten Artikeln), who=, quantity= und object-spec=. Die Mög-
lichkeiten zur Bildung von Werten sind unbeschränkt. Zum einen sind auch Eigenna-
men und Zahlen als Werte möglich, so daß bereits die Zahl der atomaren Werte über
alle Schranken wächst. Zum anderen impliziert die Möglichkeit der rekursiven Bil-
dung von Werten aus anderen die Möglichkeit, beliebig tief verschachtelte Ausdrücke
to bilden: \ldots in dem Bett im Raum des Hotels auf dem Berg...
3.3 Sprachübersetzung im Nespole! Projekt


Abbildung 6 Vereinfachte Darstellung der Komponenten, die an der Sprachübersetzung beteiligt sind, nach Lavie et al. (2001)

Besonderes Augenmerk legen wir auf das Analysemodul, das für die Transformation von natürlichsprachlichem Material in IF-Ausdrücke zuständig ist.


Im zweiten Schritt wird die Folge der Argumentbäume in Segmente unterteilt, die jeweils dem semantischen Gehalt genau einer SDU entsprechen. Zum Beispiel wird hierbei *Ja, hallo, hier ist Stephan Vogel* zerlegt in *Ja, hallo, hier ist Stephan Vogel*. Wo sich die Segmentierung nicht aufgrund technischer Restriktionen deterministisch
ergibt, wird ein einfaches statistisches Modell zur Hilfe genommen.
Der dritte Schritt generiert schließlich zu der Sequenz der Argumentbäume eines Segments einen Dialog Act, der den IF-Ausdruck vervollständigt. Dies geschieht mit Hilfe automatischer Klassifikatoren, auf die an dieser Stelle nicht näher eingegangen werden soll.
Algorithmischen Einzelheiten zum oben skizzierten Vorgehen entnehme man (Langley et al., 2002).
In den folgenden Kapiteln werden rein statistisch arbeitende Verfahren zur Generierung von IF-Ausdrücken aus natürlichsprachlichem Material vorgestellt, die eine Alternative zum oben beschriebenen Vorgehen darstellen. Der Schwerpunkt liegt dabei auf Schritt 1 (Generierung der Argumentbäume). Schritt 3 läßt sich auf einfache Weise mitbehandeln. Aus organisatorischen Gründen (Form der Trainingsmenge, S. 79) wird zu Schritt 2 vereinfachend angenommen, daß jede Eingabe genau ein Segment umfaßt, unter dem Terminus Satz ist im folgenden also stets ein Segment zu verstehen.7

Abbildung 7 Syntaxbaum über einer englischen Wortfolge. Aus den entsprechenden Nonterminalen dieses Syntaxbaums wird der IF-Unterbaum accommodation-spec = (hotel, identifiability = no, accommodation-class = three-star) generiert. Entnommen aus (Han, 2002).

7In der Evaluation ergibt sich hiezu kein Vorteil für unsere Implementierung, da hierzu auch in das grammatische System vorsegmentierte Daten eingegeben wurde.
4 Statistisch ins Interchange Format

Die Verallgemeinerungen des n-Gramm-Sprachmodells und der Übersetzungsmodelle IBM 1 und IBM 2 sollen zunächst allgemein für Baum- oder Termsprachen vorgestellt werden, bevor sie auf das spezielle Interchange Format angewendet werden, wie es im Nespole!-Projekt verwendet wird.

Um Sprachverstehen als statistische Übersetzung von natürlicher Sprache in eine rekursiv verzweigende Termssprache zu formulieren, sind nach Gleichung (1) auf Seite 20 ein Sprachmodell zur Schätzung von \( p(e) \) (\( e \) nun ein Term) sowie ein Übersetzungsmodell zur Schätzung von \( p(f|e) \) (\( e \) ein Term und \( f \) ein natürlichsprachlicher Satz) zu entwickeln.

4.1 Formale Grundlagen

Zur Hervorhebung ihres formalen Charakters bezeichnet man die als atomar betrachteten Einzelteile eines Baums als Symbole. Sie entsprechen den Wortformen in natürlichsprachlichen Sätzen.

Als Satz bezeichnet man eine endliche lineare Sequenz von Wortformen. Formal wird dies in der folgenden induktiven Definition ausgedrückt:

1. Jede Wortform \( w \) für sich genommen ist ein Satz.
2. Ist \( s \) ein Satz und \( w \) eine Wortform, so ist auch \( (s, w) \) ein Satz.

Man beachte, daß in diesem Zusammenhang ein Satz nicht notwendig ein grammatisch wohlgeformter Satz sein muß. Vielmehr ist bereits jede beliebige Kette von Wortformen ein gültiger Satz. (Diese Sprech- und Bezeichnungsweisen sind bereits aus Kapitel 2 bekannt.)

Die für Sätze etwas künstlich wirkende obige Definition läßt sich nun unmittelbar verallgemeinern zur Definition von Termen oder Bäumen:

1. Jedes Symbol \( s \) für sich genommen ist ein Term, speziell ein Blatt.
2. Jede Variable \( x \) für sich genommen ist ein Term.
3. Sind \( t_1, \ldots, t_n \) Terme und ist \( f \) ein Symbol, so ist \( t = f(t_1, \ldots, t_n) \) ein Term. In diesem Fall heißt \( f \) die Wurzel des Terms \( t \) und \( n \) heißt die Stelligkeit (oder Breite) des Terms \( t \).\(^8\) Die Terme \( t_1, \ldots, t_n \) heißen Unterterme von \( t \).

\(^8\) Man beachte, daß hier anders als in der formalen Logik üblich Stelligkeit eine Eigenschaft eines Terms – nicht des Funktionssymbols – ist, daß also zwei Terme \( f(x) \) und \( f(x, y) \) sich im gleichen Kontext nicht notwendig ausschließen.
Die obige Gegenüberstellung der Definitionen von Sätzen und Termen legt bereits nahe, in welchem Sinne Sätze als spezielle Termen aufzufassen sein werden: Dem Satz \( w_1(w_2(w_3(w_4))) \) entspricht der Term \( w_1(w_2(w_3(w_4))) \).

Nicht formal eingeführte Begriffe werden in ihrer üblichen Bedeutung verwendet. Zu einem Term \( t \) sei die Tiefe \( d = \partial t \) definiert durch die Anzahl der Knoten des längsten Pfades in \( t \). Ist \( t' \) ein (nicht notwendig direkter) Unterterm in \( t \) und \( s \) die Wurzel von \( t' \), so bezeichnet \( \partial_i(s,t) \) die Bottom-Up-Tiefe des Knotens \( s \) in \( t \) (maximale Zahl von Knoten auf einem Pfad von \( s \) zu einem Blatt des gleichen Unterbaums). Weiter heißt dann \( \partial_i(s,t) := \partial t - \partial t' \) die Top-Down-Tiefe von \( s \) in \( t \) (Zahl der Knoten auf dem Pfad von \( s \) zur Wurzel von \( t \)).

Als Ebenen in einem Term \( t \) bezeichnen wir die Menge aller Knoten gleicher Tiefe. Genauer heißt

\[
L_i(t,i) := \{ s : \partial_i(s,t) = i \}
\]

die Bottom-Up-Ebene von \( t \) zum Index \( i \) und

\[
L_i(t,t) := \{ s : \partial_i(s,t) = i \}
\]

die Top-Down-Ebene von \( t \) zum Index \( i \).

Schließlich ist der \( i \)-te Kopf eines Terms \( t \) derjenige Term \( t' = \text{top}_i(t) \), der aus \( t \) durch Ersetzung aller Symbole aus \( L_i(t,i+1,t) \) samt zugehöriger Untertermen durch verschiedene Variablen entsteht.

![Diagramm der Ebenen](image)

**Abbildung 8** Größen in einem Baum. Die Menge der grauen Knoten bilden zusammen die Bottom-Up-Ebene 1.

### 4.2 Ein Sprachmodell für Termsprachen

Der statistischen Modellierung von natürlichen Sprachen liegt die Vorstellung zugrunde, daß man eine lineare Kette von Wörtern von links nach rechts dadurch generiert, daß man in jedem Schritt das Wort hinzufügt, das unter Berücksichtigung der zuvor erstzeugten Wörter am wahrscheinlichsten ist:

\[
p(e) = p(e_1, \ldots, e_i)
\]
\[= p(e_i|e_1, \ldots, e_{i-1}) \cdot p(e_1, \ldots, e_{i-1})
\]
\[= \ldots
\]
\[= \prod_{i=1}^{l} p(e_i|e_1, \ldots, e_{i-1}).
\]
Der Generierung von Termen soll analog hierzu die Vorstellung zugrunde gelegt werden, daß jeder Teilterm dadurch generiert wird, daß zunächst rekursiv seine direkten Unterteilungen bestimmt werden, bevor unter Berücksichtigung dieser die Wurzel des Teilterms bestimmt wird. Dies führt für einen Term \( t \) mit Wurzel \( f \) und direkten Unterteilungen \( t_1, \ldots, t_r \) in direkter Entsprechung zu Gleichung (8) auf die Wahrscheinlichkeitszerlegung

\[
p(t) = p(f|t_1, \ldots, t_r) \cdot \prod_{k=1}^{r} p(t_k|t_1, \ldots, t_{k-1}).
\]

Die Tatsache, daß in der speziellen Termsprache, die wir betrachten (Abschnitt 3.2), die Reihenfolge der direkten Unterteilungen bedeutungslos für die repräsentierte Semantik ist (Kommutativität: \( f(a,b) = f(b,a) \)), rechtfertigt die heuristische Annahme, daß die auf der rechten Seite der Gleichung auftretenden bedingten Wahrscheinlichkeiten \( p(t_k|t_1, \ldots, t_{k-1}) \) unabhängig von den \( t_1, \ldots, t_{k-1} \) sind.

Diese Annahme führt auf die Form

\[
p(t) = p(f|t_1, \ldots, t_r) \cdot \prod_{k=1}^{r} p(t_k), \quad (9)
\]

in der die \( p(t_k) \) rechts sich nunmehr nach der gleichen Formel rekursiv weiter zerlegen lassen. Zu schätzen bleibt damit nur noch die Größe \( p(f|t_1, \ldots, t_r) \). Dies geschieht wie im sequentiellen Fall (Abschnitt 2.2) durch die Verwendung von relativen Häufigkeiten. An die Stelle der \( n \)-Gramme treten dabei die oben definierten Termköpfe:

\[
p(f|t_1, \ldots, t_r) \approx \frac{n(f(top_{n-1}(t_1), \ldots, top_{n-1}(t_r)))}{\sum_v n(v(top_{n-1}(t_1), \ldots, top_{n-1}(t_r)))}.
\]

Die Analogie zum sequentiellen Fall wird in Abbildung 9 durch eine Gegenüberstellung der Konzepte veranschaulicht. Während in Gleichung (9) auf die Kommutativität im Interchange Format zurückgegriffen wurde, um das Modell zu vereinfachen, geht dieses Wissen in die Schätzung der \( n \)-Gramm-Wahrscheinlichkeiten nicht ein. Bisher werden also etwa \( f(g,h) \) und \( f(h,g) \) verschieden behandelt. Ein elegantes und wahrnehmbares Hilfsmittel zur Besetzung dieses Mankos ist die Definition einer Normalform des Interchange Formats. Die Idee dabei ist, daß statt der im Corpus auftretenden \( f(g,h) \) und \( f(h,g) \) deren gemeinsame Normalform zum Training betrachtet wird.

In einer Termsprache wird eine Normalform formal durch ein noethersches und konfluentes Termersetzungssystem definiert. Ein solches System, das zwei Termen \( t_1, t_2 \) genau dann die selbe Normalform \( t' \) zuordnet, wenn sich \( t_1 \) und \( t_2 \) durch Permutation verschwisterter Unterbäume ineinander überführen lassen, ist schnell gefunden: Für jede beliebige Termordnung \( \prec \) bildet zum Beispiel der Regelsatz

\[
f(t_1, \ldots, t_i, t_{i+1}, \ldots, t_r) \rightarrow f(t_1, \ldots, t_{i+1}, t_i, \ldots, t_r) \quad \text{falls} \; t_{i+1} \prec t_i
\]

ein System mit der gewünschten Eigenschaft. Der Einsatz von Termersetzungssystemen zur Normalisierung von Interchange-Format-Ausdrücken ist dabei natürlich nicht auf die Auflösung der Kommutativität beschränkt. Prinzipiell ist auch der Einsatz komplizierter Regeln wie

\[
\text{operator} = x, [y = z_1, y = z_2, \ldots, y = z_a] \; \rightarrow \; y = (\text{operator} = x, [z_1, \ldots, z_a])
\]
denkbar. Unsere Implementierung (Kapitel 6) verwendet bisher jedoch nur ein Termersetzungssystem zur Auflösung der Kommutativität.

Zusätzlich zum Einsatz von Verfahren zur Normalisierung von Termen lassen sich Auswirkungen durch schlechte Schätzwerte ähnlich wie im sequentiellen Fall durch allgemeine Verfahren wie Interpolation und/oder Backing Off mildern. Die Techniken, die dazu in Abschnitt 2.2 vorgestellt wurden, übertragen sich ohne Veränderung.

Die vorgestellte Modellierung nach Gleichung (9) erweist sich in der praktischen Anwendung als unvorteilhaft. Weil dort ein \( p(t_q) \) als Produkt mehrerer Wahrscheinlichkeiten im allgemeinen weit kleiner sein wird als die Standardabweichung in der Ver teilung von \( p(f|t_1, \ldots, t_r) \), bevorzugs das Modell Terme von niedrigerer Stelligkeit. Diese Bevorzugung ist so stark, daß in ersten Experimenten, die dieses Modell verwendeten, nur degenerierte Bäume erzeugt wurden, die eine maximale Breite von 1 nicht überstiegen (also Listen).

Als pragmatische Alternative bietet sich an, statt des Produkts das Minimum der Untermwahrscheinlichkeiten zu verwenden. Dies führt auf das Minimummodell mit der Gleichung

\[
p(t) = p(f|t_1, \ldots, t_r) \cdot \min_{k=1}^{r} p(t_k).
\]

Mit dem Produktmodell aus Gleichung (9) teilt es die Eigenschaft, für den Spezialfall degenerierter Bäume zur Ausgangszerlegung (2) des sequentiellen Falls zu führen.

Man beachte, daß weder Gleichung (9) noch Gleichung (10) formal ein Wahrscheinlichkeitsmaß definiert: Etwa für die pathologische Termsprache \( T = \{a(b,c),d(b,b)\} \) liefern beide Modelle \( \sum_{t \in T} p(t) < 1 \). Die Maße \( p \) aus (9) und (10) sind daher allgemeiner als Bewertungsfunktionen zu verstehen.

Im folgenden wird nur das Minimummodell verwendet, sofern nicht ausdrücklich anders deklariert.

---

**Abbildung 9** Das Konzept des \( n \)-Gramms für Sequenzen (a) und Bäume (b)

### 4.3 Übersetzungsmodelle für Termsprachen

Solange ein Übersetzungsmodell nur auf Übersetzungs wahrscheinlichkeiten einzelner Wörter beruht, ohne dabei Zuordnungen und dergleichen in Betracht zu ziehen, macht es keinen Unterschied, ob man es bei der betrachteten Struktur mit einem Satz oder mit einem Term zu tun hat. Man erhält folglich ein IBM-1-Modell zur Modellierung der
Wahrscheinlichkeit $p(f|t)$ (f sequentiell, t ein Term) durch eine beliebige Indizierung der Knoten von t. Ist etwa $e_1, \ldots, e_l$ eine solche Indizierung, so überträgt sich das IBM-1-Modell wörtlich:

$$p(f|t) = \frac{1}{(l+1)m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i).$$

Höhere Modelle, in denen z. B. wie im IBM-2-Modell auch Zuordnungswahrscheinlichkeiten berücksichtigt werden, werfen dagegen die Frage nach einer adaquaten Indizierung der Knoten eines Baumes auf. Die allgemeinste Möglichkeit wäre dabei sicher die Verwendung eines im Zusammenhang mit Termersetzungssystemen gebräuchlichen Indizierungsverfahren, in dem als Index eines Knotens $k$ in einem Term $t$ der „Zugriffspfad“ von der Wurzel von $t$ bis hin zu $k$ dient. Zum Beispiel wäre dabei $(1, 2, 3)$ der Index der Wurzel des dritten direkten Unterbaums im zweiten direkten Unterbaum im ersten direkten Unterbaum von $t$. An die Stelle der Länge $l$ von $t$ trat die Äquivalenzklasse der Terme, deren Indexmenge (Menge der Indizes der Knoten eines Terms) gleich ist der Indexmenge von $t$.


Motiviert durch die Kommutativität im Interchange Format, die sich bereits im Sprachmodell (s. a.) zu Vereinfachungen ausnutzen ließ, erscheint es gerechtfertigt, bei der Indizierung die Position eines Knotens relativ zu seinen Geschwistern zu vernachlässigen und nur seine Tiefe zu berücksichtigen. Dies geschieht bei einer Modellierung nach

$$p(f|t) = \prod_{j=1}^{m} \sum_{i=1}^{l} a(i, j, m, \partial t) |L_f(t, i)| \sum_{e \in E_f(t, i)} t(f|e), \quad (11)$$

in der gewissermaßen sowohl Elemente des IBM-2-Modells (in vertikaler Richtung) als auch Elemente des IBM-1-Modells (in horizontaler Richtung) eingehen. Während die $a(i, j, m, \partial t)$ ausdrücken, mit welcher Wahrscheinlichkeit das $j$-te Wort eines Satzes mit $m$ Wörtern in einem Term $t$ einem Symbol auf Ebene $i$ zugeordnet wird, drückt der zusätzliche Faktor $|L_f(t, i)|$ aus, daß die Zuordnung mit gleicher Wahrscheinlichkeit zu jedem Symbol auf dieser Ebene geschehen kann.

Die Verwendung von $L_f(t, i)$ hat gegenüber $L_i(t, i)$ den Vorteil, daß gleiche Unterterme vom Modell gleich behandelt werden unabhängig davon, wo sie sich befinden. Zum Beispiel würden bei Verwendung von $L_i(t, i)$ im Fall $t = f(g(h, l), k(g(h, l)))$ die
beiden Auftreten des Unterterms $g(h, i)$ unterschiedlich behandelt, d.h. sich entsprechende Symbole würden in verschiedenen Ebenen erscheinen.

Gleichungen zum Training der Parameter erhält man wie beim herkömmlichen IBM-2-Modell. Wie dort sucht man nach Funktionen $t$ und $a$, die zu einer Menge vorgegebener Paare $(f, t)$ die Wahrscheinlichkeit $p(f | t)$ maximieren, unter den Nebenbedingungen, daß $\sum_e t(f | e) = 1$ und $\sum_m a(i | j, m, d) = 1$ gelten. Gesucht sind also die Stellen, an denen die partielle Ableitung von

$$h(t, a, \lambda, \mu) = p(f | t) - \sum_{e'} \lambda_{e'} \left( \sum_j (t(f | e') - 1) - \sum_j \mu_{jmd} \left( \sum_\epsilon a(i | j, m, d, \epsilon) - 1 \right) \right)$$

nach $t(f | e)$ bzw. nach $a(i | j, m, d)$ verschwindet (vgl. S. 27). Mit $L_i := L_i (i, t)$ führt dies für $t(f | e)$ auf

$$\frac{\partial}{\partial t(f | e)} h(t, a, \lambda, \mu) = p(f | t) \cdot \sum_{j=1}^m \left( \frac{\sum_{i=0}^d a(i | j, m, d)}{\vert L_i \vert} \sum_{e \in L_i} \delta(f, f_j) \delta(e, e_j) \right) - \lambda_e = 0$$

$$\implies t(f | e) = \frac{\lambda_e^{-1} p(f | t) \cdot t(f | e)}{\sum_{j=1}^m \left( \frac{\sum_{i=0}^d a(i | j, m, d)}{\vert L_i \vert} \sum_{e \in L_i} \delta(e, e_j) \right)}$$

und für $a(i | j, m, d)$ auf

$$\frac{\partial}{\partial a(i | j, m, d)} h(t, a, \lambda, \mu) = p(f | t) \cdot \sum_{j=1}^m \left( \frac{\sum_{i=0}^d a(i | j, m, d)}{\vert L_i \vert} \sum_{e \in L_i} \delta(f, f_j) \right) - \mu_{jmd} = 0$$

$$\implies a(i | j, m, d) = \frac{\mu_{jmd}^{-1} p(f | t) \cdot \frac{1}{\vert L_j \vert} \sum_{j=1}^m \sum_{e \in L_j} \delta(f, f_j) \delta(e, e_j)}{\sum_{k=0}^d \sum_{k \in \vert L_j \vert} \sum_{e \in L_k} \delta(f, f_k)}$$

Mit diesen angepaßten Counts haben die Trainingsgleichungen die gewohnte Form

$$t(f | e) = \lambda_e^{-1} \sum_{(f, t) \in C} c(f | e, f, t), \quad a(i | j, m, d) = \mu_{jmd}^{-1} \sum_{(f, t) \in C} c(i | j, m, d, f, t).$$

Die Konstanten $\lambda_e$ und $\mu_{jmd}$ dienen dabei wieder zur Normierung der Verteilungen.

Das durch Gleichung (11) beschriebene Übersetzungsmodell wird in dieser Arbeit als IBM-2-Modell für Bäume bezeichnet. In diesem Modell wird für die Zuordnungswahrscheinlichkeiten $a$ eine Abhängigkeit von Satzposition und -länge auf der einen und von Ebene und Termtiefe auf der anderen Seite angenommen. Im Falle zweier ähnlicher natürlicher Sprachen wird dies durch die Beobachtung gerechtfertigt, daß die
4.4 Decodierung

Zuordnungsfunktion $a$ häufig in weiten Teilen monoton ist, daß also weit vorne stehende Wörter häufig weit vorne stehenden und weit hinten stehende Wörter weit hinten stehenden zugeordnet werden. Derartige Zusammenhänge sind bei Bäumen nicht zu vermuten. Zwar sind die Quellsatzposition und die Termtiefe jeweils für sich bedeutsame Größen, doch die Vermutung erscheint gerechtfertigt, daß diese Größen nur schwach korreliert sind. Neben IBM 1 und IBM 2 erscheint es daher sinnvoll, ein vereinfachtes IBM 2-Modell (sIBM-2-Modell) zu betrachten, in dem $a(i|j,m,\partial t)$ nicht von $j$ und $m$ abhängt:

$$p(f|t) = \prod_{j=1}^{m} \sum_{t=1}^{\partial t} a(i|j,\partial t) \sum_{e \in L_j(t,i)} \sum_{f \in L_j(t,i)} e.$$  (12)

Das Training der $a(i|j,\partial t)$ erfolgt dabei analog zum Training der $a(i|j,m,\partial t)$ im IBM-2-Modell. In der Praxis sind die Unterschiede zwischen den drei Modellen marginal.

4.4 Decodierung


![Diagramm]

Abbildung 10 Analogon zu Abbildung 5: Decodierung von Bäumen durch Wahl einer Menge von Unterbäumen und eines neuen Symbols als Wurzel
Es erscheint zunächst abschreckend, daß in jedem Schritt statt \( n \cdot k \) neuen Hypothesen nun \( 2^n \cdot k \) neue Hypothesen erzeugt werden sollen. Die Lage wird überraschend günstig, da es zur Generierung nicht-vollständiger Bäume notwendig ist, sämtliche Hypothesen früherer Iterationen im Speicher zu behalten und als potentielle Unterbäume auf höherer Ebene mitzuverwenden. Zumindest beim Interchange Format läßt sich aber durch Einsatz von Zusatzwissen über die Termstruktur der Suchräume erheblich einschränken (s. u.). Unter anderem dadurch gelingt es, die Explosion des Suchraums relativ weit hinauszögern.

Bei der Bewertung der Hypothesen unterscheiden sich die Präfixe des sequentiellen Falls von den Untertümern im vorliegenden Fall vor allem dadurch, daß neben dem Teil „über“ dem zu bewertenden Unterbaum auch dessen Geschwister unbekannt sind, so daß zusätzlich für diese der Beitrag abgeschätzt werden muß, den sie zur späten Gesamtwahrscheinlichkeit liefern (Abb. 11). Zwar mögen spätere benachbarte Untertümern zum Zeitpunkt der Bewertung bereits dekodiert sein und sich in der Menge der Hypothesen befinden, doch ist zum Zeitpunkt der Bewertung offen, welche Untertümern zu einem späteren Zeitpunkt benachbart sein werden.

Abb. 11 Bewertung einer Hypothese: Während im sequentiellen Fall nur der Beitrag der noch nicht dekodierten Wörter abgeschätzt ist, sind bei der Bewertung eines dekodierten Untertümens zusätzlich die Beiträge der Nachbarregionen abzuschätzen.

Die Bewertungsfunktion \( \tau \) für Gesamthypothesen bleibt davon zunächst unberührt. An die Stelle der Zielsatzlänge \( l \) tritt die Termtiefe \( d \), im übrigen Überträgt sich die Funktion aus Gleichung (7) wörtlich vom sequentiellen Fall:

\[
\tau(H) := p(H) \cdot \sum_{d=0}^{\infty} p(d|m) \prod_{j=1}^{m} \sum_{i=0}^{d} a(i,j,m,d) \tau(i,j,H).
\]

Die nötigen Änderungen schlagen sich allein in der Definition von \( \tau(i,j,H) \) nieder, durch das der Beitrag der \( i \)-ten Ebene von \( H \) zur Generation des \( j \)-ten Quellworts abgeschätzt wird. Mit \( |L_j(H,i)| := \max \{1, |L_j(H,i)| \} \) sei

\[
\tau(i,j,H) := \sum_{b=|L_j(H,i)|}^{\infty} \frac{p(b|i,d)}{b} \left[ \sum_{e_j \in L_j(H,i)} \frac{\tau(e_j)}{p(f_j|e_j)} + (b - |L_j(H,i)|) \sum_v p(v) \right].
\]

Hierbei durchläuft \( b \) alle möglichen Breiten des späteren Baumes auf Ebene \( i \), die zugehörige Wahrscheinlichkeit \( p(b|i,d) \) kann vom Sprachmodell bereitgestellt werden.

\( \text{Kernzeichen sind damit Bäume, deren Blätter verschiedene Top-Down-Tiefen haben dürfen} \)
4.4 Decodierung

Der normierende Faktor $1/b$ drückt die Gleichverteilung der Zuordnungswahrscheinlichkeiten innerhalb der $i$-ten Ebene aus und entspricht dem Faktor $1/|L_s(t,i)|$ in Gleichung (11). Die eckige Klammer liefert dann den Beitrag der $i$-ten Ebene für den Fall, daß diese Ebene im fertigen Baum $b$ Elemente umfassen wird. Die vorhandenen Symbole $e_j$ tragen jeweils $t(f_j,e_j)$ bei, jedes der übrigen $b - |L_s(H,i)|$ noch unbekannten Symbole liefert einen Beitrag von $\sum_e p(v)p(f_j|v)$. 

Man beachte, daß obige Definition von $\tau(i,j,H)$ für degenerierte Bäume mit der Definition des sequentiellen Decoders (vgl. S. 29) zusammenfällt. Dort gilt $p(b|i,d) = 1$ im Fall $b = 1$ und $i \leq d$, sonst $p(b|i,d) = 0$. Weiter ist $|L_s(H,i)| = 1$ genau dann, wenn das $i$-te Wort $e_i$ bereits decodiert ist, ansonsten ist $|L_s(H,i)| = 0$. Da der Wert der eckigen Klammer also nur für den Fall $b = 1$ relevant ist, verschwindet wegen $|L_s(H,i)| \in \{0,1\}$ genau einer der beiden Summanden, was auf Seite 29 durch die Fallunterscheidung ausgedrückt wird.

Wie bereits angekündigt, sind bei einer Implementierung des Decoders spezielle Vorkenungen nötig, die den Suchraum in geeigneter Weise beschränken. In der Implementierung dieser Arbeit geschieht dies auf verschiedenerlei Art und Weise.

Zunächst wählt der Decoder aus dem gesamten Lexikon eine Menge $\{e_1, \ldots, e_k\}$ von Symbolen (Argumente und atomare Werte) aus, für die der Wert $\sum_{j=1}^{m} t(e_j|f_j)$ $(i = 1, \ldots, k)$ maximal wird. Die $t(e_j|f_j)$ entstammen einem umgekehrte trainierten IBM-1-Modell. Die Anzahl $k$ wird in Abhängigkeit von der Länge $m$ von $f$ bestimmt. Die Erfahrung zeigte, daß bei Wahl von $k = 5 + 4 \cdot \log m$ mit einer Wahrscheinlichkeit von 87,46% ein Symbol der Referenz unter den ausgewählten Symbolen $e_1, \ldots, e_k$ ist.\footnote{Bei den übrigen Symbolen handelt es sich meist um Teile von Spezialkonstruktionen, oder es befinden sich unter den $e_j$ ein Symbol mit semantisch ähnlicher Funktion.} Innerhalb des Decoders wird daher $\{e_1, \ldots, e_k\}$ als Lexikon der Zielsprache verwendet.


Schließlich läßt sich auch das Wissen aus der Spezifikation des Interchange Formats zu bedeutsamen Effizienzsteigerungen ausnutzen. Ein speziell entwickelter Algorithmus (beschrieben im Anhang ab Seite 101) zählt nur diejenigen Mengen von Hypothesen auf, die gemeinsam als Unterbäume einer neuen Hypothese in Frage kommen. Dadurch werden ungültige Bäume, in denen ein Argument oder ein Wert mehrfach auftritt, nicht generiert, und wegen der Kommutativität werden keine zwei Mengen
aufgezählt, die sich durch Permutation ihrer Elemente ineinander überführen lassen. Nicht allgemein durch Iteration lösbare ist die Frage nach Lizensierungen der Unterbäume sowie die Einschränkung, daß jeder Rahmen über nur einen Kopf verfügen darf. Hypothesenmengen, die diesen Anforderungen nicht genügen, werden aussortiert.

Anders als die beiden anderen Mechanismen zur Effizienzsteigerung ist bei der Ausnutzung von Wissen über die internen Zusammenhänge im Interchange Format zu bemerken, daß sie keine Suchfehler verursachen kann. Es ist garantiert, daß in den Teilen des Suchraums, die hierbei abgeschnitten werden, keine korrekten Lösungen zu finden sind.

4.5 Besonderheiten beim Interchange Format

In den bisherigen Ausführungen war allgemein von Termsprachen die Rede, auch wenn in die Entwicklung der statistischen Modelle und des Decoders bereits Spezialwissen aus der Definition des Interchange Formats eingeflossen ist. Jedoch blieben zwei spezielle Konstruktionen bisher unberücksichtigt: der Dialog Act sowie die Wertelisten (Abschnitt 3.2).

Die Behandlung von Wertelisten läßt sich sehr leicht durch Einführung zusätzlicher Funktionssymbole realisieren. So wird etwa der Term

\[ \text{room-spec} = [(\text{single}, \text{quantity} = 2), (\text{double}, \text{quantity} = 1)] \]

von Seite 34 intern als

\[ \text{room-spec} = (\text{set} = (\text{empty} = (\text{single}, \text{quantity} = 2), \text{empty} = (\text{double}, \text{quantity} = 1))) \]


Die Behandlung von Listen stellt während des Decodings ein besonderes Problem dar. Da hier Unterterm zu set= mehrfach auftreten können und Einschränkungen durch Lizensierungen erst im weiteren Verlauf möglich werden, würde eine Behandlung der Listen nach dem üblichen Verfahren eine enorme Vergrößerung des Suchraums nach sich ziehen. Statt also beim Decoding set= und empty= wie alle anderen Argumente zu behandeln, werden Listenkonstrukte besonders behandelt. Nachdem alle übrigen Hypothesen einer Iteration gebildet wurden, werden für jedes Argument zusätzliche Hypothesen erzeugt. Sind etwa \( a = b, a = c, a = (d = e, f), a = (g = h) \) alle existierenden Hypothesen zum Argument \( a = \), so werden daraus zusätzlich die Hypothesen

\[ a = [b,b], a = [b,c], a = [b,(d = e, f)], \ldots, a = [b,c,(d = e, f),(g = h)] \]

Neben den Listen stellt auch der Dialog Act eine spezielle Konstruktion im Interchange Format dar, die sich nicht unmittelbar auf allgemeine Term sprachen übertragen läßt. Mit einer gewissen Berechtigung läßt sich zwar der Standpunkt vertreten, der Speech Act (also z. B. give-information) sei die „wirkliche“ Wurzel des Terms, während die Konzepte bereits die erste Top-Down-Ebene des Terms bilden. Problematisch an dieser Sichtweise ist aber, daß sich in einem bestehenden Ausdruck die Argumente der oberen Ebene nicht mehr eindeutig einem Konzept zuordnen lassen. In

\[ c: \text{give-information} + \text{reservation} + \text{trip(who = i)} \]

ist zum Beispiel unklar, ob das Argument who= eher dem Konzept \( +\text{reservation} \) oder \( +\text{trip} \) zuzuordnen ist. (Es wird von beiden lizensiert.) Zudem spricht dagegen, daß die Reihenfolge der Konzepte im Dialog Act für seine Bedeutung relevant ist, während unsere Termmodelle von einer kommutativen Term sprache ausgehen.


Die Decodierung eines vollständigen IF-Ausdrucks ist dann ein zweistufiger Prozeß. Zuerst wird wie in Abschnitt 4.4 beschrieben der Argumentbaum erzeugt. Der Baum decoder übergibt die Liste der \( n \) besten Hypothesen an den Wurzle decoder. Die Generierung einer Wurzel gestaltet sich dann als besonders einfach. Aufgrund der Einschränkungen, die die Spezifikation bezüglich der Abfolge von Konzepten macht, ist es möglich, die am besten bewertete Wurzel unter all jenen auszuwählen, die mit einem der Argumentbäume zusammenpassen. Sind \( t_i \), \( i = 1, \ldots, n \) die decodierten Argumentbäume und ist \( r_j \) die zu \( t_i \) bestimmte Wurzel, so wird am Ende das Paar \( (r_j, t_i) \) als Übersetzung von \( f \) zurückgegeben, für das das Produkt

\[ p(r_j) \cdot p(f|r_j) \cdot p(t_i) \cdot p(f|t_i) \]

maximal wird.
5 Projektion von Übersetzungsmodellen

Die Behandlung des Sprachverstehens mit Hilfe statistischer Methoden verlangt, wie in den vorherigen Kapiteln deutlich geworden ist, ein bilinguales Corpus, anhand dessen mit Hilfe der diversen Trainingsalgorithmen die Wahrscheinlichkeitsverteilungen der jeweiligen statistischen Modelle bestimmt werden. Im vorliegenden Fall bedeutet das, daß gesammeltes Sprachmaterial mit Interlingua zu annotieren ist. Für diese Arbeit sind Experten nötig.


Dieses Kapitel stellt allgemeine Verfahren zur Projektion der Verteilungen der ersten beiden IBM-Modelle vor. Sie werden formuliert für das Problem, ein Modell $F/E$ zur Übersetzung von $F$ nach $E$ wird erzeugt durch Projektion eines Modells $F/G$ und eines Modells $G/E$ auf das Paar $F/E$.


Prinzipiell bestehen bei der Projektion von Übersetzungsmodellen drei Ursachen, die einen Qualitätsverlust gegenüber einem direkt trainierten Modell bewirken können. Zunächst gehen zwei Corpora statt eines einzigen in die Schätzung der Parameter ein, die naturgemäß beide verraucht sind. Zwar wurden bei sehr kleinen Trainingsmengen Effekte beobachtet, die darauf hin deuten, daß der Umweg über die dritte Sprache zur Glättung von schlecht geschätzten Verteilungen beitragen kann (vgl. Abschnitt 7.4), doch werden im allgemeinen zwei Corpora zusammen stärker rauschen als ein einzel-
ner (Varianz ist additiv). Ferner sind Probleme zu erwarten, wenn der Ausschnitt der Sprache $G$ in den Trainingsmengen für $F/G$ und $G/E$ stark unterschiedlich ist.


Als drittes wären schließlich Verluste durch heuristische Annahmen im Projektionsverfahren denkbar. Wie unten zu zeigen sein wird, gelingt für die Modelle IBM 1 und 2 die Projektion ohne Zuhilfenahme heuristische Annahmen, so daß Verluste dieser Art in unserem Fall ausgeschlossen sind.

Projektion von Übersetzungsmodellen bedeutet Projektion von Wahrscheinlichkeitsverteilungen. In der vorliegenden Situation gilt es also, für die Modelle IBM 1 und 2 die Verteilung $t(f|e)$ aus den Verteilungen $t(f|g)$ und $t(g|e)$ zu gewinnen. Für das IBM-2-Modell sind zusätzlich die Zuordnungsvariablen $a(i,j,m,l)$ zu gewinnen. Wie üblich bezeichnen $i, j$ Positionen und $l, m$ die Längen von $e, f$. Zusätzlich wird $k$ als Position in $g$ und $n$ für die Länge von $g$ verwendet. Stattdessen $l$ und $a$ wird einheitlich $p$ für Wahrscheinlichkeiten verwendet.

5.1 Projektion von IBM-1-Modellen

Die Wahrscheinlichkeiten $t$ lassen sich als Übergangswahrscheinlichkeiten in einer Markov-Kette auffassen. Die Wörter $f, g$ und $e$ sind dann Zustände dieser Kette und $p(f|g)$ beschreibt die Wahrscheinlichkeit, als nächstes den Zustand $f$ zu betreten, wenn man sich zur Zeit im Zustand $g$ befindet. Der Übergangszustande enthält also zu jedem Wort der drei Sprachen $F$, $G$ und $E$ einen Zustand und zwischen je zwei Wörtern verschiedener Sprachen eine Kante. Die Gewichte der Kanten zwischen Wörtern $f, g$ und $g, e$ sind nach Voraussetzung bekannt, zu ermitteln sind daraus die Gewichte der Kanten zwischen Wörtern $f, e$.

Unter Verwendung der Chapman-Kolmogorov-Gleichung erhält man

$$\sum_g p(g)p(f|g)p(g|e) = \sum_g p(g) \frac{p(f,g)}{p(g)} \frac{p(g,e)}{p(e)} = \frac{1}{p(e)} \sum_g p(f,g)p(g,e) = \frac{p(f,e)}{p(e)} = p(f|e).$$

Sobald neben $p(f|g)$ und $p(g|e)$ auch die Unigramm-Wahrscheinlichkeiten $p(g)$ zur
Verfügbar stehen, erhält man die gesuchte Verteilung \( p(f|e) \) also via
\[
p(f|e) = \sum_{g} p(g)p(f|g)p(g|e).
\] (14)

Die Beschaffung der Unigramm-Wahrscheinlichkeiten \( p(g) \) während des Trainings von \( p(f|g) \) und/oder \( p(g|e) \) stellt dabei kein besonderes Problem dar.

### 5.2 Projektion von IBM-2-Modellen

Zur Projektion von IBM 2 Modellen sind neben den Übersetzungswahrscheinlichkeiten \( t \) zusätzlich die Zuordnungswahrscheinlichkeiten \( a(i|k,n,l) \) und \( a(k|j,m,n) \) nach \( a(l|j,m,l) \) zu übertragen. Dieser Prozeß gestaltet sich vergleichbar mit der Projektion der Übersetzungswahrscheinlichkeiten als komplizierter, da mit \( k \) und \( n \) nunmehr zwei Variablen zu eliminieren sind. Damit dies gelingt, werden einige zusätzliche Verteilungen benötigt.

Neben den trivial zu bestimmenden Wahrscheinlichkeiten \( p(n), p(m|n), p(n|m), p(n|l) \) und \( p(l|m) \) wird zusätzlich die Zuordnungswahrscheinlichkeit \( p(j|n,m) \) benötigt. Dies erlaubt es, mit \( n \) und \( m \) zu arbeiten, indem man parallel zum Training von \( p(k|j,m,n) \) das Paar \( F/G \) ein vereinfachtes IBM-2-Modell für \( G/F \) trainiert, bei dem die Abhängigkeit von \( k \) im Training unberücksichtigt bleibt.

In einem ersten Schritt sind die Längenwahrscheinlichkeiten zu projizieren. Dies geschieht analog zur Projektion der Übersetzungswahrscheinlichkeiten nach
\[
p(m|l) = \sum_{n} p(n)p(m|n)p(n|l)
\] (15)
\[
p(l|m) = \sum_{n} p(n)p(l|n)p(n|m).
\] (16)

Die \( p(n|l) \) erlauben es dann, in einem zweiten Schritt die Hilfsverteilung \( p(j|m,l) \) zu erzeugen. Wegen
\[
\frac{1}{p(m|l)} \sum_{n} p(n)p(m|n)p(n|l)p(j|n,m)
= \frac{p(l)}{p(m,l)} \sum_{n} p(n)p(m,n)p(n,l)p(j,m,n)
= \frac{1}{p(m,l)} \sum_{n} p(n,l)p(j,n,m) = \frac{p(j,m,l)}{p(m,l)} = p(j|m,l)
\]
erhält man die \( p(j|m,l) \) offenbar über
\[
p(j|m,l) = \frac{1}{p(m|l)} \sum_{n} p(n)p(m|n)p(n|l)p(j|m,n).
\] (17)

Alle benötigten Größen stehen zur Verfügung.

Die nun vorliegenden Hilfsgrößen \( p(l|m) \) und \( p(j|m,l) \) erlauben schließlich im dritten Schritt zusammen mit \( p(k|j,m,n) \) und \( p(i|k,n,l) \) die Erzeugung der gesuchten Vertei-
lungen \( p(i|j,m,l) \). Die Rechnung

\[
\frac{1}{p(l|m)p(j|m,l)} \sum_{i} \left( \frac{p(n)p(n|m)p(j|m,n)p(l|n)}{p(m)p(m,n)p(n)} \cdot \sum_{k} p(k|j,m,n)p(k,n,l)p(i,k,n,l) \right)
\]

\[
= \frac{1}{p(l|m)p(j|m,l)} \sum_{i} \left( \frac{p(n)p(m,n)p(j,m,n)p(l,n)}{p(m)p(m,n)p(n)} \cdot \sum_{k} p(k,j,m,n)p(k,n,l)p(i,k,n,l) \right)
\]

\[
= \frac{1}{p(m)p(l|m)p(j|m,l)} \sum_{k} p(k,j,m,n)p(k,n,l) \cdot \frac{p(i,j,m,n,l)}{p(j,m,l)} = p(i,j,m,l)
\]

führt auf die Projektionsformel

\[
p(i|j,m,l) = \frac{1}{p(l|m)p(j|m,l)} \sum_{i} \left( \frac{p(n)p(n|m)p(j|m,n)p(l|n)}{p(m)p(m,n)p(n)} \cdot \sum_{k} p(k,j,m,n)p(k,n,l)p(i,k,n,l) \right).
\] (18)

Alle in ihr auftretenden Größen sind bekannt, nachdem \( p(l|m) \) und \( p(j|m,l) \) zuvor ihrerseits durch Projektion nach (16) bzw. (17) bestimmt wurden.

Man beachte, daß für ein durch projektion gewonnenes Modell \( t(f|e), a(i|j,m,l) \) auch die nötigen Hilfsverteilungen \( p(m|l), p(l|m), a(j|m,l) \) zur Verfügung stehen, so daß sich das beschriebene Projekionsverfahren iteriert anwenden läßt, z. B. zur Überbrückung mehrerer Sprachen (Abb. 13).

**Abbildung 13** Doppelte Anwendung der Projektion, um ein Modell \( F/C \) (Französisch nach Chinesisch) aus drei Modellen \( F/G, G/E \) und \( E/C \) zu erzeugen.
6 Das System JTrans

Um den Erfolg der entwickelten Verfahren in praktischen Anwendungssituationen beurteilen zu können, wurde im Rahmen der Diplomarbeit ein Softwaresystem entwickelt, mit dessen Hilfe sich das Training der verschiedenen Modelle, das Decodieren sowie verschiedene automatisierte Bewertungen durchführen lassen. Beim Entwurf dieses Systems stand das Ziel im Vordergrund, durch eine flexible Architektur zunächst eine Grundlage zu schaffen, die den Benutzer in die Lage versetzt, mit minimalen Aufwand neue Ideen zu Forschungszwecken zu implementieren. Laufzeit- und Speichereffizienz spielten dagegen eine untergeordnete Rolle.

6.1 Grundkonzept des Systems


```plaintext
> a = "Hallo"
> b = "Welt"
> a + " " + b
Hallo Welt
> exit()
```


Neben den Methoden stellt das System einige globale Funktionen zur Verfügung. Zu ihnen gehört z.B. das im obigen Beispielprogramm bereits aufgetretene exit() zum Verlassen des Programms. Funktionen werden in Bibliotheken definiert, die mit Hilfe von load((Name)) in das System eingeladen werden können. Stets eingeladen ist die spezielle Bibliothek Std, von der neben anderen die Funktionen exit() (zum Beenden des Programms), load((Bibliothek)) (zum Einladen weiterer Bibliotheken), time() (für Laufzeitmessungen), read((Dateiname)) (zum Einlesen einer Befehlsdatei) und type((Objekt)) (zur Bestimmung des Typs eines Objekts) definiert werden.

Das vorliegende Kapitel versucht einen ersten Einblick in die vom System bereitgestellte Infrastruktur zu geben. Dabei wird auch auf die Implementierung der Verfahren
aus Kapitel 4 und 5 eingegangen. Für weitere Einzelheiten zur Architektur und zu den vorhandenen Methoden sei auf den Anhang dieser Arbeit (vgl. Seite 91) sowie auf die in elektronischer Form vorliegende Dokumentation (vgl. auch Seite 105 dieser Arbeit) verwiesen.

6.2 Basisfunktionalität

Neben der puren Benutzerschnittstelle werden vom eigentlichen System auch häufig gebrauchte Datenstrukturen sowie Unterklassen von Concept zur Verfügung gestellt, die neben rein architektonischen Klassen auch Standardimplementierungen typischer Sprach- und Übersetzungsmodelle umfassen.

Die wichtigsten dieser Klassen sind im Klassendiagramm in Abbildung 14 dargestellt.


Corpus-Objekte repräsentieren multilingual Corpora, d. h. Aufzählungen von Tupeln einander entsprechender Sätzen in verschiedenen Sprachen. Typischerweise bezieht das Corpus seine Daten aus einer Textdatei, in der jede Zeile ein Tupel enthält und die Dimensionen innerhalb einer Zeile durch das Steuerzeichen ',' abgetrennt sind. Die erste Zeile einer Datei definiert dabei die Dimension des gesamten Corpus. Das
Corpus stellt seine Daten in Form von Iteratoren zur Verfügung. So liefert die Methode `getStringIterator()` einen `StringIterator` zurück, der alle Einträge einer bestimmten Dimension durchläuft. Wenn zuvor mittels `setLexicon()` ein Lexicon für eine Dimension definiert wurde, steht zusätzlich die Methode `getPhraseIterator()` zur Verfügung, die einen `PhraseIterator` für die gewünschte Dimension zurückliefer. Iteratoren für verschiedene Dimensionen durchlaufen dabei die Tupel in der gleichen Reihenfolge, so daß die i-ten Elemente verschiedener Iterator garantiert zum gleichen Tupel gehören. Corpora lassen sich via `add()` addieren (Konkatenation) und via `mul()` multiplizieren (kartesisches Produkt).


Implementierungen von `LanguageModel` und `TranslationModel` können optional eine Addition zur Verfügung stellen, die über die Methode `add()` aufgerufen wird.\(^{11}\) Die Addition erlaubt die Zusammenfassung von Modellen, die auf verschiedenen Corpora trainiert wurden, zu Gesamtmodellen.


\(^{11}\) An der Benutzerschnittstelle kann man auch das Infix-Symbol `*` zur Addition verwenden. Intern wird $a + b$ wie $a: add(b)$ behandelt.

```java
> c = new Corpus("train.corp")
> src = new NGramLM(1) // trivial model
> src:learn(c, 0)
> tgt = new NGramLM(3) // trigram model
> tgt:learn(c, 1)
> ibm1 = new Ibm1TM(src, tgt)
> ibm1:learn(5, c, 0, 1)
> ibm2 = new Ibm2TM(src, tgt)
> ibm2:learn(5, c, 0, 1, ibm1)
> ibm2:setDecoder(new WWDecoder())
> ibm2:trans(new Corpus("test.in"), 0, "test.out")
> exit()
```

Weitere Beispielssituationen dieser Art zusammen mit vorbereiteten Datensätzen sind auf der beiliegenden CD (vgl. S. 105) zusammengestellt.

### 6.3 Realisierung der Übersetzung ins Interchange Format

Das Grundsystem stellt zunächst nur eine allgemeine Umgebung für Experimente mit statistischen Verfahren zur Verfügung. Es ist nicht speziell auf unsere Interessen zugeschnitten, viel mehr bilden die Implementierungen der Verfahren aus Kapitel 4 ein eigenständiges Paket, das das Basissystem um die nötige Funktionalität erweitert. Eine Übersicht über dieses Paket zeigt Abbildung 15.

Abbildung 15 Die Klassen des Pakets **nespole**. Grau hinterlegte Klassen gehören zu anderen Paketen.

Sprach- und Übersetzungsmodelle bestehen analog zur *IFPhrase* jeweils aus einem eigenen Modell für die Wurzel (*Root*) und den Argumentbaum (*Tree*). So ist *IFTM* die Zusammenfassung eines *RootTM* und eines *TreeTM* und *IFLM* analog die Zusammenfassung eines *RootLM* und eines *TreeLM*. Da die Wurzel wie eine gewöhnliche *Phrase* behandelt werden kann, wie in Abschnitt 4.5 ausgeführt wurde, fungieren die Klassen *RootLM* und *RootTM* als Adapter zu „normalen“ Modellen. Ihre einzige Aufgabe ist es, eintreffende *IFPhrase*-Objekte wie etwa

\[
e : \text{give-information} + \text{disposition} + \text{trip} \text{(visit-spec = vacation, disposition = desire)}
\]

termaxzunehmen und in der Form

\[
\text{give-information} + \text{disposition} + \text{trip}
\]

an das sequentielle Modell weiterzuleiten. Dagegen wird in *TreeLM* und *TreeTM* nur der Argumentbaum betrachtet, also etwa

\[
\text{root} = (\text{visit-spec = vacation, disposition = desire}).
\]

*TreeLM* und die Unterklassen von *TreeTM* implementieren dabei die Modelle, die in den Abschnitten 4.2 und 4.3 erarbeitet wurden.

Aus Platzgründen nicht in Abbildung 15 enthalten ist die Architektur um die Decoder. Sie folgt dem gleichen Schema: Ein *IFDecoder* ist ein *Decoder* und besitzt einen *RootDecoder* sowie einen *TreeDecoder*. Eine Übersetzungsanfrage an den *IFDecoder* wird zunächst an dessen *TreeDecoder* weitergereicht, der eine Bestenliste erzeugt. Diese wird dann zusammen mit dem Quellsatz und dem *IFTM* an den *RootDecoder* weitergereicht, der zu jedem Baum der Bestenliste eine Wurzel erzeugt. Schließlich wählt der *IFDecoder* aus allen Wurzel/Baum-Paaren eines aus und gibt eine entsprechende *IFPhrase* zurück. Wie *TreeTM* ist auch *TreeDecoder* eine abstrakte Klasse. Eine Implementierung des Decoders aus Abschnitt 4.4, passend zu *TreeBm2TM*, findet sich in einer Klasse *TreeBm2Decoder*.

```java
> c = new Corpus("train.corp")
> daFile = "da.db.lsp"
> argsFile = "nespole-arguments.db.lsp"
> valsFile = "nespole-values.db.lsp"
> iflex = new nespole.IFLexicon(daFile, argsFile, valsFile)
> src = new NGramLM(1)
> src:learn(c, 0)
> tgt-root = new nespole.RootLM(new NGramLM(4))
> tgt-root:setLexicon(iflex)
> tgt-root:learn(c, 1)
> tgt-tree = new nespole.TreeLM(3)
> tgt-tree:setLexicon(iflex)
> tgt-tree:learn(c, 1)
> tgt = new nespole.IFILM(tgt-root, tgt-tree)
> ibml1-root = new nespole.RootTM(new Ibm1TM(src, tgt-root))
> ibml1-root:learn(5, c, 0, 1)
> ibml2-root = new nespole.RootTM(new Ibm2TM(src, tgt-root))
> ibml2-root:learn(5, c, 0, 1, ibml1-root)
> ibml2-root:setRootDecoder(new nespole.RootIbm2Decoder())
> ibml1-tree = new nespole.TreeIbm1TM(src, tgt-tree)
> ibml1-tree:learn(5, c, 0, 1)
> ibml2-tree = new nespole.TreeIbm2TM(src, tgt-tree)
> ibml2-tree:learn(5, c, 0, 1, ibml1-tree)
> ibml2-tree:setTreeDecoder(new nespole.TreeIbm2Decoder())
> tm = new nespole.IPTM(ibml2-root, ibml2-tree)
> tm:trans(new Corpus("test.in"), 0, "test.out")
> exit()
```

Inhaltlich entspricht die obige Sitzung weitgehend dem Beispiel aus Abschnitt 6.2, so daß auf weitergehende Erläuterungen an dieser Stelle wohl verzichtet werden kann. Für eine vollständige Dokumentation der zur Verfügung stehenden Klassen und Methoden sei abschließend noch einmal auf den Anhang dieser Arbeit (S. 91) sowie auf elektronisch verfügbares Material auf der CD (S. 105) verwiesen.
7 Evaluation


Standardmäßig ist A die Trainingsmenge. Wo mit anderen Systemen verglichen wurde, mußte aus Kompatibilitätsgründen auf B trainiert werden. Stets ist C die Testmenge (d.h. ihre Elemente werden z.B. übersetzt), sie kann bezüglich A und B als ungeschätzt betrachtet werden. Tabelle 1 zeigt statistische Kenngrößen der drei Mengen, eine ausführliche Analyse der Corpora findet man im Anhang ab Seite 79.

<table>
<thead>
<tr>
<th>Menge</th>
<th>Sprache</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>Sätze</td>
<td></td>
<td>3885</td>
<td>3885</td>
<td>2427</td>
</tr>
<tr>
<td>versch. Sätze</td>
<td></td>
<td>2135</td>
<td>2113</td>
<td>1395</td>
</tr>
<tr>
<td>einm. Sätze</td>
<td></td>
<td>1970</td>
<td>1951</td>
<td>1283</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50,70%</td>
<td>50,21%</td>
<td>52,86%</td>
</tr>
<tr>
<td>Tokens</td>
<td></td>
<td>17434</td>
<td>18248</td>
<td>11236</td>
</tr>
<tr>
<td>Wörter</td>
<td></td>
<td>1558</td>
<td>1213</td>
<td>1196</td>
</tr>
<tr>
<td>einm. Wörter</td>
<td></td>
<td>755</td>
<td>489</td>
<td>566</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48,46%</td>
<td>40,30%</td>
<td>47,34%</td>
</tr>
</tbody>
</table>

Tabelle 1 Statistische Kenngrößen zu den Datensätzen A, B und C

Für die Experimente wurde für die Argumentbäume ein Trigramm-Modell und für die Wurzeln ein Tetragramm-Modell verwendet. Als Übersetzungsmodell kommt standardmäßig ein IBM-2-Modell zum Einsatz, das wie im Beispiel auf Seite 58 trainiert wurde.

7.1 Vergleich verschiedener Zuordnungsalgorithmen

Seit Brown et al. (1993) steht außer Frage, daß in natürlichen Sprachen Zuordnungen einander entsprechender Wörter innerhalb eines Satzpaars bei geeigneter Trainingsmenge mit großer Treffsicherheit richtig automatisch vorhergesagt werden können. Es scheint eine inhärente Eigenschaft natürlicher Sprachen zu sein, daß sich zwischen semantisch äquivalenten Sätzen in vielen Fällen sinnvolle Wort-zu-Wort-Zuordnungen finden lassen. Angewandt auf eine Kunstsprache wie das Interchange Format bedarf
das Konzept der Zuordnung jedoch einer erneuten Motivation. Macht es Sinn, Wörter eines natürlichsprachlichen Satzes etwa den Symbolen eines Argumentbaums zuorden zu wollen?

Die nähere Betrachtung von Paaren zeigt, daß gewisse Wörter (vor allem semantikarme Füllwörter) keine Entsprechungen im Argumentbaum finden, daß andererseits manche Symbole in einem Argumentbaum (vor allem in den höheren Ebenen) zu umfassend sind, als daß man sie einem einzelnen Wort zuordnen könnte. Zuordnungen von deutschen Testtsätzen zu den zugehörigen Argumentbaumen, die per Hand bestimmt wurden, zeigten, daß sich nur etwa 58,32% der deutschen Wörter sinnvoll Symbolen des Argumentbaums zuordnen ließen.\footnote{Hier und im folgenden werden nur nichttriviale Paare betrachtet.} Betrachtet man die Zuordnungen von den gleichen deutschen Sätzen zu ihren englischen Übersetzungen, so kommt man auf eine Quote von 87,98%. Neben den schon genannten Füllwörtern ist diese Diskrepanz an manchen Stellen auch auf eine gewisse „Unvollständigkeit“ der Annotation zurückzuführen. Wenn auf die Codierung von Nuancen verzichtet wird, fehlt Material, das eine Zuordnung möglich machen könnte. Als Argumentbaum zu meine Frau hat mir hier (Segment g051ak.33.4, vgl. S. 79) findet man in der Datenbank etwa

\[ \text{root} = \text{(to-whom} = i, \text{person-spec} = \text{(whose} = i, \text{spouse, sex} = \text{female}).] \]

In diesem Baum findet man zu hat und hier keine Entsprechung, weil die Information dieser Wörter im Baum nicht codiert ist. Eine erschöpfendere Codierung würde durch Einfügen der zusätzlichen Unterbäume e-time = previous für hat und location = here für hier erreicht, die Material zur Zuordnung bereitstellen würden.\footnote{Es ist dabei unklar, ob die Beschränkung auf zentrale semantische Einheiten gegenüber einer vollen Abdeckung zu bevorzugen ist. Zum einen könnte nämlich die Codierung sämtlicher Details Schwierigkeiten bei der späteren Generierung versuchen. Zum anderen kann man die Frage stellen, ob z.B. das hier im Beispiel wirklich semantiktragend ist, oder eher eine sprachinterne Funktion hat.} Als drittes schließlich ist zu bedenken, daß Wurzel und Argumentbaum getrennt behandelt werden. Teils lassen sich nämlich längere Phrasen einem einzigen Symbol der Wurzel zuordnen, die dann im Argumentbaum selbst keine Entsprechung mehr finden. Ein Beispiel hierfür ist Wie sieht das aus mit der Verpflegung? (Segment g051ak.14.2), in der Datenbank codiert durch

\[ \text{c} : \text{request-action + inform + meal(food-spec} = \text{(food, identifiability} = \text{yes}).] \]

Hier werden die ersten fünf Wörter sowie das Fragezeichen am Schluß durch die Symbole request-action und +inform dargestellt, während nur der und Verpflegung eine Entsprechung im Argumentbaum finden (nämlich yes und food).

Trotz der genannten Probleme erscheint eine Zuordnung von Wörtern zu Symbolen insbesondere für die semantisch bedeutsamen Teile eines Paars möglich zu sein. Es bleibt festzustellen, ob diese Zuordnungen auch automatisch gefunden werden können. Zu diesem Zweck wurden die Zuordnungen betrachtet, die die Termversionen der Modelle IBM 1 und IBM 2 (vgl. Abschnitt 4.3) liefern. Genauer wurde ein \( f_j \) des Quellsatzes beim ersten Modell dem Symbol \( e_k \) mit

\[ i = \arg \max_k (f_j | e_k) \]
und beim zweiten Modell dem Symbol \( e_i \) mit
\[
i = \arg \max_k a(\hat{\theta}_i, e_k | j, m, d) t(f_j | e_k)
\]
zugeordnet. Die so generierten Zuordnungen wurden dann begutachtet, um zu erfahren, wie nah sie an die zuvor von Hand erstellten heranreichen. Die Zahl der sinnvoll zugeordneten Wörter wurde dazu zur Zahl der Wörter in Relation gesetzt, die bei der Handzuordnung sinnvoll zugeordnet werden konnten. Das Ergebnis zeigt einen überraschend kleinen Unterschied zwischen den beiden Modellen: Dem IBM-1-Modell gelangen 86,67% der möglichen Zuordnungen und dem IBM-2-Modell 87,73%.\(^{14}\)


\[\text{Abbildung 16} \quad \text{Zuordnung zwischen einem deutschen Satz und dem zugehörigen Argumentbaum a) nach dem IBM-1-Modell, b) nach dem IBM-2-Modell.}\]

\(^{14}\)Die entsprechenden Zahlen für zwei natürliche Sprachen (Deutsch nach Englisch statt Deutsch nach Interchange Format) lauten 89,35% (IBM 1) bzw. 82,59% (IBM 2).
Zusammenfassend läßt sich feststellen, daß beide Zuordnungsverfahren in der Lage sind, für die Wörter eines Satzes, die sich überhaupt sinnvoll zuordnen lassen, gute Zuordnungen auch zu finden. Zwischen den Verfahren lassen sich leichte qualitativa Unterschiede erkennen, was die Zuordnungen zu inneren Knoten angeht. Wo solche Zuordnungen nötig sind, hat das IBM-1-Modell zwar eine größere Trefferquote, doch spricht für das IBM-2-Modell, daß es in kritischen Fällen eher zum tieferen Knoten tendiert, und so eine bessere Grundlage für den Decoder schafft. Zudem liegt das IBM-2-Modell auch quantitativ leicht im Vorteil.


<table>
<thead>
<tr>
<th></th>
<th>Deutsch/IF</th>
<th>Deutsch/Englisch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimale Quote</td>
<td>58,32%</td>
<td>87,97%</td>
</tr>
<tr>
<td>IBM 1 Quote</td>
<td>r · 86,67%</td>
<td>r · 89,35%</td>
</tr>
<tr>
<td>IBM 2 Quote</td>
<td>r · 87,73%</td>
<td>r · 82,59%</td>
</tr>
</tbody>
</table>

Tabelle 2 Vergleich der Trefferquoten in den Modellen IBM 1 und IBM 2

### 7.2 Qualität der erzeugten Terme

Die Qualität der erzeugten Terme wurde durch zwei Vergleiche bestimmt. Im einen Vergleich werden die generierten Terme verglichen mit jenen, die von Experten von Hand erzeugt wurden. Der andere Vergleich betrachtet die generierten Terme im Vergleich zu jenen, die das offizielle grammatik arbeitende System (Lavie et al., 2001) liefert. Da die Interchange Format Version, mit der das grammatische System arbeitet, von der Version abweicht, in der die Annotation der Testmenge vorliegt, mußte für die beiden Vergleiche eine unterschiedliche Trainingsmenge zugrunde gelegt werden. Die angegebenen Zahlen sind daher nur innerhalb desselben Kontexts vergleichbar.


Der große Anteil der Modellfehler ist dagegen nur bedingt als wirkliche Schwäche des Modells zu deuten. Oftmals sind die Terme in der Datenbank einfach reichhaltiger (also größer) und werden gegenüber kleineren Termen, die weniger Details beinhalten, vom Modell benachteiligt. Als Beispiel hierfür diene das Segment *gibt es da Ermäßigung für Kinder?* (g047ak.20.2). In der Datenbank wird diesem Segment die Annotation

\[
c: \text{request-information + existence + price}
\]

\[
\text{price-spec = (for-whom = (quantity = plural, child), discount),}
\]

\[
\text{location = there)}
\]

zugeordnet. Sie wird vom Modell mit der Wahrscheinlichkeit \( p = 6,63 \cdot 10^{-41} \) bewertet. Der Decoder liefert für dasselbe Segment die Darstellung

\[
c: \text{request-information + existence + price}
\]

\[
\text{price-spec = (quantity = plural, discount),}
\]

\[
\text{for-whom = (quantity = plural, child)),}
\]

die vom Modell mit \( p = 7,81 \cdot 10^{-37} \) bewertet wird. Hierin wird die Ortsangabe *da* nicht codiert und fälschlich *Ermäßigungen* statt *Ermäßigung* verstanden. Das Beispiel zeigt ferner, daß von der Statistik (insbesondere vom Sprachmodell, vgl. Abschnitt 4.2) breite Strukturen gegenüber tiefen bevorzugt werden. Wegen dieser geringen Unterschiede wird man die gefundene Hypothese aber kaum als schlecht bezeichnen können.

Sinnverfälschende Fehler, die sich meist auch in größeren Abständen zwischen den Bewertungen niederschlagen, treten vor allem auf, wenn Wörter der Eingabe unbekannt sind. Der Abstand

\[
d := |\log p(\text{Referenz}) - \log p(\text{Hypothese})|
\]

beträgt in der gesamten Testmenge im Durchschnitt 8,36 Punkte. Wenn man die Sätze aus der Betrachtung herausnimmt, die unbekannte Wörter enthalten, so fällt dieser Wert auf 3,51 herab. Da sich hinter unbekannten Wörtern meist ein beachtlicher Teil
der Semantik verbirgt, ist einzusehen, daß Sätze mit unbekannten Wörtern nur schwierig zu behandeln sind. (Beispiel: Das System sieht das Segment g051ak.14.5 als daß die (unk) (unk) werden? . Es fehlen genau die semantiktragenden Wörter mittags und verpflegt.) Wenn genügend Kontext zur Verfügung steht, kann die inhärente Unschärfe von statistischen Verfahren allerdings in Einzelfällen dazu führen, daß das fehlende Wort richtig erraten wird, etwa so, wie auch ein Mensch aus dem Kontext abschätzen kann, welches das fehlende Wort sein könnte. Zum Segment hin ich da mit der (unk) verbunden (g051ak.2.1, das fehlende Wort ist Touristenauskunft) wird zum Beispiel der Term
c : request-introduce-self
   (who = tourist_information_center, communication-mode = phone)
generiert, der exakt mit der Referenz der Datenbank übereinstimmt. In der Mehrzahl der Fälle gelingt dies jedoch nicht so glücklich.

Beim Vergleich mit dem grammatischen System fällt zunächst auf, daß bei fast der Hälfte der Sätze (48,45%) nicht feststellbar ist, welcher der von den beiden Systemen generierten Sätze besser ist als der andere. In 30,93% findet man gar eine exakte Übereinstimmung (dies hauptsächlich bei den trivialen Sätzen).

Überraschenderweise gelang es dem statistischen Ansatz aber teilweise sogar, bessere Terme zu erzeugen als das grammatische System. In der Testmenge gilt dies für etwa 13,92% der Fälle. So liefert das regelbasierte System den Ausdruck c : acknowledge als Repräsentation von sehr schön (z. B. Segment g047ak.14.1), während der statistische Ansatz die reichhaltigere Darstellung

c : give-information + concept
   (concept-spec = (modifier = (nice, intensity = intense)))

lieferd, die vom englischen Generator perfekt in very good überführt wird, während mit okay das andere Resultat zwar nicht falsch, aber doch weiter vom Original entfernt ist.

<table>
<thead>
<tr>
<th>Vergleich</th>
<th>besser</th>
<th>schlechter</th>
<th>ähnlich</th>
<th>gleich</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 2 vs. Grammatik, gesamt</td>
<td>13.92%</td>
<td>37.63%</td>
<td>48.45%</td>
<td>30.93%</td>
</tr>
<tr>
<td>IBM 2 vs. Grammatik, Wurzel</td>
<td>23.71%</td>
<td>24.23%</td>
<td>52.06%</td>
<td>42.27%</td>
</tr>
<tr>
<td>direkt trainiert vs. projiziert</td>
<td>27.32%</td>
<td>14.95%</td>
<td>57.75%</td>
<td>24.23%</td>
</tr>
</tbody>
</table>


7.2 Qualität der erzeugten Terme

wird es bei diesen Sätzen für den statistischen Ansatz immer schwieriger Schritt zu halten.
Sobald aber Konstruktionen auftreten, die der Grammatik unbekannt sind, erreicht der statistische Ansatz mitunter auch für nichttriviale Sätze bessere Ergebnisse als die Grammatik. Dies zeigt sich eindrucksvoll am Beispiel des Segments *wo Kinder auch ganz gut fahren können* (Segment g047ak.7.6). Konstruktionen wie diese werden offenbar von der verwendeten Version der Grammatik nicht abgedeckt, denn diese liefert den Ausdruck

\[
\text{c : request-information + feasibility + view + object}
\]

\[
\text{(location = question),}
\]

\[
\text{object-spec = (whose = (child, quantity = plural),}
\]

\[
\text{focalizer = additive,}
\]

\[
\text{modifier = good)).}
\]

Die Subjunktion *wo* wurde hier offensichtlich als Fragewort aufgefaßt (im Speech-Act request-information und im Untertermin location = question), ferner wurde verstanden, daß es „auch“ (focalizer = specific) um ein „gutes“ (modifier = good) Objekt „der Kinder“ (whose = (child, quantity = plural)) gehe. Das Konzept +feasibility drückt aus, daß es um eine Fähigkeit geht („fahren können“), +view läßt sich nicht sinnvoll zuordnen.
Der statistische Ansatz liefert hier das deutlich bessere Resultat

\[
\text{c : give-information + feature + activity}
\]

\[
\text{(who = (focalizer = additive, child),}
\]

\[
\text{activity-spec = (skiing, modifier = (intensity = medium, good))).}
\]

Es drückt in etwa aus, daß „auch Kinder“ (who = ...) „ganz gut“ (modifier = ...) „(Ski) fahren“ (skiing). Aus Sicht des Argumentbaums ist dies nahezu ideal, ein Schonheitsfehler ist das +feature anstelle von +feasibility in der Wurzel.\(^{15}\)

Zu einer überraschenden Erkenntnis gelangt man, wenn man den Vergleich auf die generierten Dialog Acts beschränkt, die Argumentbäume also unberücksichtigt läßt. In den meisten Fällen (52,06\%) ist subjektiv kein Qualitätsunterschied feststellbar, d. h. beide Ausgaben sind gleich gut oder beide sind gleich schlecht. 42,27\% der gelieferten Dialog Acts stimmen gar exakt überein. Wo ein Qualitätsunterschied feststellbar ist, wird der bessere Dialog Act zu gleichen Anteilen vom grammatischen und vom statis-tischen System geliefert: 23,71\% der statistischen sind besser als die entsprechenden grammatischen und 24,23\% der grammatischen sind besser als die entsprechenden statistischen. Dies zeigt, daß die beiden Verfahren bei der Generierung des Dialog Acts etwa gleichlauf liegen.

\(^{15}\)Man könnte vielleicht vermuten, daß das Trainingscorpus bereits einen ähnlichen Satz enthält. Dies ist aber nicht der Fall: Zum Beispiel treten auch und Kinder nur zwölf mal gemeinsam in einem Satz des Corpus auf, wobei sich das auch nur dreimal wirklich auf die Kinder bezieht. Das Bigram ganz gut tritt zweimal auf und wird einmal davon in der Annotation gar nicht codiert.
Es läßt sich sagen, daß es dem statistischen Ansatz besser gelingt, Negationen zu erkennen und Fragen von Aussagen zu unterscheiden. So liefert die Grammatik für kann man das Skigebiet als familienfreundlich bezeichnen? (g051ak.11.3) den Dialog Act

c : give-information + feasibility + activity

und zu ich hab' noch keinen Ort. (g047ak.5.2) den Dialog Act

c : give-information + existence + attraction.

Statistisch bekommt man in diesen Fällen

c : request-information + feasibility + activity

bzw.

c : negate-give-information + existence + object.

Dagegen ist die Grammatik treffsicherer, wenn es um Fragen der Disambiguierung geht, z. B. wenn in das wäre sehr schön (g051ak.47.2) festzustellen ist, daß das ein Pronomen und kein Artikel ist. Die Grammatik liefert hier mit

с : give-information + feature + object

ein besseres Ergebnis als das von der Statistik gelieferte с : give-information + feature.

Erstaunlich ist, daß es der Statistik vereinzelt sogar gelingt, Strukturen zu erkennen, die sich über mehrere Wörter erstrecken, obwohl die Modelle bisher ja nur auf der Wortebene arbeiten. So wird dann gleich noch 'ne Frage speziell zum Alpin (g051ak.47.5) dargestellt als

с : introduce-topic + information-object,

verglichen mit dem allgemeineren с : give-information + concept der Grammatik ist dies ein sehr gutes Ergebnis.

Zusammenfassend läßt sich sagen, daß mit dem statistischen Verfahren überraschend gute Interchange-Format-Ausdrücke generiert werden können. Die Qualität des Dialog Acts ist dabei etwas höher als die des Argumentbaums.

7.3 Vergleichende Humanevaluation


Um die Kompatibilität zum Generierer sicherzustellen war die Verwendung der Trainingsmenge B nötig, beide statistischen Systeme verwendeten diese Menge zum Training.
7.3 Vergleichende Humanevaluation


<table>
<thead>
<tr>
<th>(1) rein statistisch</th>
<th>40,30%</th>
<th>22,68%</th>
<th>37,03%</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) IF-grammatisch</td>
<td>18,90%</td>
<td>36,34%</td>
<td>44,76%</td>
</tr>
<tr>
<td>(3) IF-statistisch</td>
<td>15,12%</td>
<td>30,15%</td>
<td>54,73%</td>
</tr>
<tr>
<td>(4) Projektion</td>
<td>19,78%</td>
<td>22,68%</td>
<td>57,56%</td>
</tr>
</tbody>
</table>

☐ perfect ☐ okay ☐ bad

**Abbildung 17** Durchschnittliche Bewertung der englischen Übersetzung durch verschiedene Evaluatoren. Angegeben sind jeweils die Anteile der Übersetzungen, die als perfekt (links), okay (mitte) und bad (rechts) bewertet wurden. Zu (4) vgl. Abschnitt 7.4.

In den End-To-End-Vergleich geht neben der Performanz der eigentlichen Generierung der Interchange-Format-Ausdrücke auch die Qualität des regelbasierten Generierers ein. Zwar ist die Entwicklung eines Generierers wesentlich unkomplizierter als die Entwicklung einer Analysegrammatik, doch ist nicht davon auszugehen, daß der vorhandene Generierer optimal ist.

Daß er nicht optimal ist, zeigt sich insbesondere bei der Behandlung verrauchter und unvollständiger Ausdrücke. Die auftretenden Effekte seien anhand zweier Segmente exemplarisch dargestellt.

Zum Segment Skifahren ist gut (g047ak.6.2) generiert die Grammatik den Ausdruck

\[
c : \text{give-information} + \text{feature} + \text{activity} \\
\quad (\text{activity-spec = skiing, feature = (modifier = good)})
\]

der vom Generierer perfekt in Skiing would be good, übersetzt wird. Der statistische
Ansatz liefert mit

\[
  c : \text{give-information + feature} \\
  (\text{activity-spec = skiing, feature = (modifier = good)}),
\]

(19)
ein geringfügig anderes Ergebnis, doch mit diesem kommt der Generierer weit we-\nniger gut zurecht. Er überführt es in *It is good*, und ignoriert die Information, daß
es ums Skifahren (activity-spec = skiing) geht. Das Ergebnis ist also „bad", obwohl
die Interchange-Format-Repräsentation zumindest „okay“ ist. Man beachte dabei, daß
der Ausdruck in (19) legal ist, da das Argument activity-spec= bereits vom Konzept
+feature lizensiert wird.

Als zweites Beispiel betrachte man das Segment *wo Kinder auch ganz gut fahren
können* (g047ak.7.6), das bereits auf Seite 65 diskutiert wurde. Aus dem Ausdruck

\[
  c : \text{give-information + feature + activity} \\
  (\text{who = (focalizer = additive, child),} \\
  \text{activity-spec = (skiing, modifier = (intensity = medium, good)))},
\]

den der statistische Ansatz liefert, macht der Generierer *It is somewhat good skiing. *
Auch dieses Ergebnis enthält nicht die gesamte Information, die im IF-Ausdruck co-
diert ist. Man würde eher ein Ergebnis der Form *Children do skiing somewhat good,
too.* erwartet.

Da Grammatik und Generierer parallel entwickelt wurden, liegt die Vermutung nah,
daß der Generierer mit grammatisch erzeugten Ausdrücken besser zurecht kommt als
mit den Ausdrücken, die der statistische Ansatz liefert. Zu erklären wäre dies dadurch,
daß in die Entwicklung des Generierers Wissen eingeflossen ist über die Struktur der
Ausdrücke, die die Grammatik generiert.

Eine nähere Betrachtung zeigt jedoch, daß dies wohl nicht der Fall ist: Bei 8,25% der
grammatisch generierten und 8,76% der statistisch generierten Ausdrücke verarbei-
tet der Generierer wesentliche Teile der zur Verfügung stehenden Information nicht.
Man kann also annehmen, daß der Generierer die Ausdrücke der Grammatik nicht
oder nur minimal bevorzugt. Weiter ist davon auszugehen, daß der Abstand zwischen
den Interlingua-basierten Systemen und dem rein statistischen System (1) zu einem
erheblichen Teil auch vom Generierer verschuldet wird.

Um den Einfluß des Generierers auf die Qualität der Übersetzung besser beurteilen
zu können, wäre es interessant zu wissen, welche Ergebnisse er auf idealen von Hand
geschriebenen Ausdrücken liefert. Ein solcher Vergleich war jedoch in der aktuellen
Situation nicht möglich, da ja die idealen Referenzen in einer anderen Version des
Interchange Formats vorliegen als der Generierer erwartet.

Zusammenfassend ist festzustellen, daß der Qualitätsunterschied zwischen den Aus-
drücken, die von der Grammatik und der Statistik generiert werden, in der End-to-
End-Betrachtung im wesentlichen erhalten bleibt. Die Übersetzungsqualität der inter-
linguabasierten statistischen Übersetzung liegt nur wenig unter der der etablierten Sy-
steme, was bei Berücksichtigung des großen Entwicklungsfortschritts dieser Systeme
bereits als Nachweis für den Erfolg des statistischen Ansatzes gelten kann.
7.4 Verluste durch Projektion


Da ferner das Projektionsverfahren an sich verlustfrei ist, weil zur Verkettung der Wahrscheinlichkeitsverteilungen für die verwendeten Modelle geschlossene Formeln angegeben werden konnten, sind die auftretenden Qualitätsverluste allein auf ungefähre Schätzungen der Ursprungsverteilungen sowie auf inhärente Sprachdifferenzen zurückzuführen (vgl. Kapitel 5).

Ein Qualitätsverlust von Übersetzungsmodellen schlägt sich in einer höheren Perplexität nieder. Der Logarithmus der Perplexität läßt sich intuitiv als Abstandsmäß auf Sprachen interpretieren. Dieser Abstand hängt dabei von Trainingsmenge und vom statistischen Modell ab, außerdem läßt sich ein so definierter „Abstand“ formal nicht als Metrik auffassen, da die dazu nötigen Voraussetzungen (etwa \(d(x,x) = 0\)) allenfalls approximativ erfüllt werden.


Es interessiert nun, wie sich ein projiziertes Modell im Vergleich zu einem direkt trainierten Modell verhält. Dazu wurden zunächst die von beiden Modellen nach Training auf der Menge \(A\) generierten Ausdrücke miteinander verglichen. Ähnlich wie in Abschnitt 7.2 wurde dazu ermittelt, wie viele Terme eines Modells subjektiv besser sind als die entsprechenden Termen des anderen Modells. Es zeigt sich dabei, daß bei 57,75% der Hypothesen kein Qualitätsunterschied feststellbar ist, Gleichheit gilt bei 24,23%.
Bei den übrigen Termen erweist sich erwartungsgemäß das auf Deutsch/IF trainierte Modell als besser. So erscheinen 27.32% der von diesem Modell gelieferten Ergebnisse besser als die Entsprechungen des projizierten Modells, in immerhin 14.95% der Fälle ist das Ergebnis des projizierten Modells besser.


Ein besonders gutes Ergebnis liefert das projizierte Modell z. B. für das Segment also ’n ’drei Sterne sollt’ es schon sein. (Segment g051ak.24.3). Den hierzu generierten Ausdruck

\[ c : \text{give-information} + \text{feature} + \text{object} \]

\[ (\text{who} = \text{pronoun}, \]

\[ \text{feature} = (\text{accommodation-spec} = (\text{accommodation-class} = \text{three_star}))) \]

verwandelt der Generierer in The three star one would be good. Verglichen mit den Ergebnissen aller anderen Systeme ist dies hervorragend (vgl. S. 87). Eine deutlich bessere Qualität gegenüber dem direkt trainierten Modell erreicht das projizierte Modell jedoch überdies nur noch bei den Segmenten g047ak.7.6 und g051ak.3.5, und so ist fraglich, ob damit der hohe Anteil an perfekten Übersetzungen ausreichend erklärt werden kann. Eher ist zu vermuten, daß ein häufig auftretendes triviales Segment zufällig besser vom projizierten Modell abgedeckt wird und die Bewertung übermäßig stark positiv beeinflußt. Ein Kandidat hierfür ist möglicherweise das zwölfmal auftretende Segment ja, das mit dem projizierten Modell perfekt zu Yes. übersetzt werden kann, während die anderen interlinguabasierten Systeme das Ergebnis Okay. liefern, das von einigen Evaluatoren auch nur als „okay“ bewertet wird.

Abschließend sei auf ein Phänomen hingewiesen, das den Einfluß struktureller Sprachdifferenzen bei der Projektion verdeutlicht. Es tritt im Zusammenhang mit der englischen Konstruktion have to in Erscheinung, durch die ein Zwang ausgedrückt wird. Im Interchange Format kann dies durch ein Konzept +obligation codiert werden. Da have auch in der Bedeutung haben verwendet wird, ergibt sich durch die Projektion eine falsche Bewertung zwischen dem deutschen haben und +obligation. Man betrachte dazu etwa den Term, den das projizierte Modell zum Segment den Lauf hab’ ich schon (g051ak7.1) generiert:

\[ c : \text{request-information} + \text{obligation} + \text{existence} + \text{object} \]

\[ (\text{property} = \text{already}, \text{experiencer} = \text{i}). \]

Ohne den Umweg über das Englische wäre das Auftreten von +obligation nur schwer zu erklären.
7.5 Einfluß der Corpusgröße

Die Ergebnisse dieses Abschnitts bestätigen die Erwartungen über die auftretenden Phänomene bei der Projektion von Übersetzungsmodellen: Die Qualitätsverluste sind gering und liegen im wesentlichen in nicht modellierten strukturellen Unterschieden der natürlichen Sprachen begründet. In Einzelfällen erzielt ein projiziertes Modell bessere Ergebnisse als ein vergleichbares direkt trainiertes Modell.


7.5 Einfluß der Corpusgröße

Bei der Arbeit mit statistischen Verfahren hängen die Resultate in besonderem Maße auch von den verwendeten Trainingsdaten ab. Oft ist man daher versucht zu behaupten, größere Trainingsmengen würden zu besseren Ergebnissen führen. Die Trainingsmengen, die für die Untersuchungen dieser Arbeit zur Verfügung stehen, sind mit 2 427 bzw. 3 885 Paaren in der Tat nicht besonders groß, so daß die Frage gerechtfertigt erscheint, ob man sich von mehr Trainingsmaterial noch Qualitätsverbesserungen versprechen kann. Der vorliegende Abschnitt zeigt Indizien auf, die diese These zu stützen scheinen.


Daß unsere Übersetzungsmodelle vermutlich übertrainiert sind, versucht Abbildung 19 zu verdeutlichen. Sie stellt die Perplexitäten von Trainings- und Testmenge nach jeder Trainingsiteration gegenüber. Exemplarisch wird dabei das Training eines Übersetzungsmodells für die Wurzel betrachtet. Das Diagramm zeigt, daß die Perplexität der Trainingsmenge in den späteren Iterationen noch fällt, während sich für die Testmenge keine Verbesserungen mehr erzielen lassen. Tatsächlich ist es sogar so, daß die Perple-

\[16\] Vogel and Tribble (2002) untersuchen spezielle Techniken für statistische Übersetzer im Falle geringer Trainingsdaten und betrachten dazu eine Menge von 3 182 Paaren.
xitäten der Testmenge zum Ende hin wieder zu steigen beginnen. Offenbar befindet sich der Konvergenzprozess demnach bereits in einer e-Umgebung um das Optimum des Corpus, die das wahre Maximum nicht mehr umfasst: Das Modell wird übertrainiert.

![Abbildung 19 Vergleich der Perplexitäten von Trainings- und Testmenge eines Übersetzungsmodells für den Dialog Act. Die vertikale Achse ist logarithmisch.](image)

Wenn man den Anteil der akzeptablen Übersetzungen zugrunde legt, ist das grammatische System mit 55,24% um etwa einen Faktor 1,2210 besser als das System, das statistisch die Interechange Format Ausdrücke generiert (45,28% akzeptable Ergebnisse, vgl. Abb. 17 in Abschnitt 7.3)

Um abzuschätzen, in wieweit dies auf die Größe des Corpus zurückzuführen ist, wurde das System auf verschiedene große Teilmengen der Menge A trainiert und die jeweils resultierenden Übersetzungsergebnisse der Menge C mit den idealen Referenzen verglichen. Durch Extrapolation dieser Werte soll abgeschätzt werden, welche Verbesserungen durch Verwendung größerer Corpora bei Verwendung der vorgestellten Modelle noch zu erwarten sind.

Als Abstandsmaß zwischen Hypothesen und Referenzen wurde eine der Situation angepaßte Definition der Wortfehlerrate verwendet. Ist $t_H$ eine Hypothese und $t_R$ eine Referenz, so wird für jeden Pfad $p_R \in t_R$ der Pfad $p_H \in t_H$ bestimmt, der die kleinste Editierdistanz $e(p_R, p_H)$ zu $p_R$ hat.\textsuperscript{17} Durch

$$\text{wer}(t_H, t_R) := \frac{1}{|t_R|} \sum_{p_R \in t_R} \min_{p_H \in t_H} e(p_R, p_H)$$

wird damit ein normiertes Fehlermaß definiert. Da Fehler im oberen Teil des Baumes implizit stärker gewichtet werden als Fehler in der Nähe der Blätter, ist nicht unbeeindruckt, da es sich bei Fehlern auf höheren Ebenen typischerweise um schwerere Fehler handelt als bei falscher Wahl von Blättern.

Abbildung 20 zeigt die durchschnittlichen Fehlerraten $\text{wer}(t_H, t_R)$ beim Training auf verschiedenen großen Präfixen der Menge A. Da zum Vergleich mit dem grammatischen System die kleinere Trainingsmenge $B$ verwendet wurde, ist die Fehlerrate der Teilmenge zu vergleichen, die die gleiche Größe wie $B$ hat (etwa 62,47% der Größe von A).

\textsuperscript{17}Ein Baum $t$ wird hier modelliert als eine Menge von Pfaden. Der Dialog Act wird wie ein Pfad behandelt.
Zur Extrapolation sei angenommen, daß sich das Abklingen der Fehlerrate durch die Funktion

\[ f: [0, \infty) \to \mathbb{R}, \quad f(x) := \alpha + (1 - \alpha)e^{-\lambda x} \]

für geeignete \( \alpha \in [0, 1] \) und \( \lambda \geq 0 \) beschreiben läßt. Mit Hilfe numerischer Verfahren findet man, daß der mittlere quadratische Abstand von \( f \) zu den Meßwerten für die Wahl von \( \alpha = 0.3763 \) und \( \lambda = 3.2809 \) minimal wird. Der Funktionsgraph von \( f \) für diese Parameter ist in Abbildung 20 den Meßwerten überlagert. Die Abbildung zeigt, daß \( f \) den Verlauf der Meßwerte relativ gut modelliert. Die Abbildung enthält ferner die Assymptote \( x \to \alpha \), gegen die \( f \) für \( x \to \infty \) konvergiert.

Um mit dem grammatischen System gleichzuziehen, wäre eine Fehlerrate \( r \) von

\[ r = \frac{1}{1,2210} f(0.6247) = 35.06\% \]

d. h. eine Reduktion um 23.18\%, nötig. Wegen \( \lim_{x \to \infty} f(x) = \alpha \) und \( f(x) > \alpha (x \geq 0) \) liegt die maximale erreichbare Reduktion bei \( 1 - \frac{\alpha}{f(0.6247)} = 17.55\% \).

Diese Analyse erlaubt die Folgerung, daß die Corpusgröße einen wesentlichen Anteil am Abstand zum grammatischen System hat. Nach den vorliegenden Zahlen sollte sich durch eine Verdopplung der Trainingsmenge der Vorsprung der Grammatik von 22.10\% auf

\[ 1 - \frac{f(2 \cdot 0.6247)}{1,2210 f(0.6247)} = 10.27\% \]

fast halbieren. Quantitative Aussagen dieser Art können auf der Grundlage der gemessenen Werte nicht als verläßliche Voraussagen angesehen werden. Trotzdem zeigen die Überlegungen dieses Abschnitts zumindest qualitativ, daß das Potential der Modelle trotz ihrer Einfachheit mit den verwendeten Trainingsmengen nicht voll ausgeschöpft werden konnte.
8 Ergebnisse und offene Fragen

Das geringe linguistische Wissen, das von den Übersetzungsmodellen von IBM verwendet wird, erlaubt nicht nur die Anwendbarkeit dieser Modelle auf beliebige Paare natürlicher Sprachen, sondern auch auf Kunstsprachen, deren Struktur sich von der Struktur natürlicher Sprachen erheblich unterscheidet. Im vorliegenden Fall wurden die statistischen Modelle derart erweitert, daß sie nicht auf einfache Wortsequenzen (Sätze) beschränkt bleiben sondern auch auf allgemeinere Strukturen wie Bäume (Terme) angewendet werden können.


Ein erheblicher Teil der Arbeitszeit wurde in die Entwicklung eines Softwaresystems investiert, das eine flexible, erweiterbare und interaktive Umgebung zur Arbeit mit Verfahren der statistischen Übersetzung bereitstellt. Mit Hilfe dieser Entwicklungsumgebung gelang im Rahmen der Diplomarbeit eine Implementierung der vorgestellten Modelle und Algorithmen, deren Performanz mit der eines etablierten grammatischen System für die gleiche Anwendungssituation vergleichbar ist.

Ein Verfahren zur Projektion von Übersetzungsmodellen wurde entwickelt, um Übersetzungsmodelle zur Übersetzung einer natürlichen Sprache in Interlingua bereits auf der Grundlage bilingualer natürlichsprachlicher Corpora zu gewinnen, für deren eine Sprache ein Modell zur Übersetzung in Interlingua vorhanden ist. Das erbringt die Annotation von gesammeltem Sprachmaterial mit Interlingua. Für die IBM-Modelle 1 und 2 läßt sich für die Projektion eine geschlossene Formel angeben, die ohne heuristische Annahmen auskommt. Dadurch wird sichergestellt, daß der Qualitätsverlust durch die Projektion einzig durch das Rauschen der Corpora sowie durch unmodelliert strukturelle Unterschiede der Sprachpaare hervorgerufen wird.


Außerdem beobachtet man, daß von der Spezifikation des Interchange Formats großzügiger lizensiert wird als eigentlich nötig. Strengere Einschränkungen würden den Suchraum für den Decoder verkleinern, so daß dieser tiefer in die abgeschnittenen Bereiche blicken kann. Kuriose Kanten wie conjunction = car, die man gelegentlich beobachtet, und die vermutlich am ehesten für ein späteres Scheitern des Generierers verantwortlich gemacht werden müssen, könnten durch Löschen der entsprechenden Lizenzierungen sehr leicht vermieden werden.

Abschnitt 7.5 zeigte, daß bereits einfache Modelle ihr Potential auf dem derzeitigen Trainingsmengen nicht voll ausschöpfen können. Beim Arbeiten mit komplizierteren Modellen steigt die Zahl der zu trainierenden Parameter, so daß bei unveränderten Trainingsdaten die Menge der Daten pro zu schätzenden Parameter fällt. Es sollten daher auch Anstrengungen unternommen werden, die darauf abzielen, das vorhandene Material besser auszunutzen. Bei der Behandlung sequentieller Sprachen bietet sich hierzu der Einsatz von Wortklassen an (vgl. z. B. Vogel et al., 2000). Dabei betrachtet man statt einer Sequenz \( f \) typischerweise die Sequenz \( \mathcal{A}(f) \) der Wortklassen \( \mathcal{A}(f_1), \ldots, \mathcal{A}(f_m) \) von \( f_1, \ldots, f_m \) und übersetzt diese in eine Sequenz \( \mathcal{B}(\epsilon) \) von Zielsprach-Wortklassen: \( \mathcal{B}(\epsilon_1), \ldots, \mathcal{B}(\epsilon) \). Mit Hilfe einer explizit bestimmten Zuordnung \( a \) werden aus den Wortklassen \( \mathcal{B}(\epsilon) \) dann die tatsächlichen Wörter \( e_i \) bestimmt. Eine Möglichkeit zur Bestimmung bilingualer Wortklassen wird von Och (1999) vorgeschlagen. Ziel ist es, die Trainingsdaten auf einem abstrakteren Niveau zu betrachten, um sie auf diese Weise besser auszunutzen zu können.

Eine direkte Anwendung dieses Vorgehens auf Argumentbäume scheint schwierig, da die verschiedenen Ebenen im Argumentbaum bereits verschiedene Abstraktionsstufen beinhalten und daher zu befürchten ist, daß sich die Wahrscheinlichkeitsmasse einfach auf einer höheren Ebene konzentriert oder – schlummer – gleichmäßig auf mehrere Ebenen verteilt. Während z. B. die Zuordnung zwischen der Phrase „hier ist Stephan Vogel“ (g047ak.1.3) und dem zugehörigen Term

\[
\text{root} = \text{(who = (given-name = name-stephan, family-name = name-vogel), communication-spec = phone)}
\]

unproblematisch ist (Stephan zu name-stephan und Vogel zu name-vogel), ist eine Zuordnung zwischen „hier ist (first-name) (last-name)“ zu

\[
\text{root} = \text{(who = (given-name = (first-name), family-name = (last-name)), communication-spec = phone)}
\]

schwieriger, da z. B. given-name= und (first-name) semantisch nicht mehr zu unterscheiden sind.
Als Alternative kommt aber die Verwendung von Wortklassen vor dem eigentlichen Training zur Vorverarbeitung der Trainingsmenge in Frage. Vorstellbar wäre eine automatisierte Corpusvergrößerung, etwa indem man für jedes vorhandene Paar \((f, e)\) der Trainingsmenge mit \(f = (f_1, f_2)\) und \(e = (e_1, e_2)\) einige der Paare \((f', e')\) mit \(f' = (f_1', f_2')\) und \(e' = (e_1', e_2')\) hinzufügt, für die \(A(f_j) = A(f'_j)\) und \(B(e_i) = B(e'_i)\) gilt.


Zum Nespole Corpus


Die Daten wurden geringfügig aufgearbeitet und zu zwei Trainingsmengen A, B sowie einer Testmenge C zusammengestellt. Die Trainingsmenge A besteht aus den Dialogen e002cp, e003ap, e005cp, e008cp, e015ap, e017ap, e033ap, e047cp, e709wa, e709wb, e710wa, e710wb und e913ab mit englischem Original und g002ct2, g004ct2, g005at2, g006at2, g010at2, g011ct2, g012at2, g017ct2, g018ct2, g020at2, g023ck2, g024ak2, g025ck2, g028ct2, g031ck2, g032ck2, g038ak, g039ck, g044ak, g045ck, g050ak, g055ak und g999ck mit deutschem Original. Die Annotationen in dieser Menge genügen der Spezifikation vom Mai 2002.


Von den deutschen Wörtern (Tokens) in C treten 7,649% nicht in der Menge A und 9,223% nicht in der Menge B auf. Die unbekannten Wörter verteilen sich im ersten Fall auf 27,83% der Sätze und im zweiten Fall auf 29,89% der Sätze. Ferner treten 38,65% der deutschen Sätze in C bereits in A und 38,14% der Sätze in B auf. (Dies gilt vor allem für triviale Sätze, s. u.)
Die entsprechenden Daten für die englische Dimension lauten wie folgt: In C sind bezüglich A 5,026% und bezüglich B 6,387% der Worte unbekannt, und diese Wörter verteilen sich auf 19,07% bzw. 22,16% der Sätze. Es treten 39,69% der englischen Sätze in C bereits in A und 38,14% der Sätze bereits in B auf.

Weitere statistische Kenngrößen der Mengen A, B und C sind in der folgenden Tabelle zusammengestellt.

<table>
<thead>
<tr>
<th>Menge</th>
<th>Sprache</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D</td>
<td>E</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>Sätze</td>
<td>3 885</td>
<td>3 885</td>
<td>2 427</td>
<td>2 427</td>
</tr>
<tr>
<td>versch. Sätze</td>
<td>2 135</td>
<td>2 113</td>
<td>1 395</td>
<td>1 395</td>
</tr>
<tr>
<td>einm. Sätze</td>
<td>1 970</td>
<td>1 951</td>
<td>1 283</td>
<td>1 290</td>
</tr>
<tr>
<td></td>
<td>50,70%</td>
<td>50,21%</td>
<td>52,86%</td>
<td>53,15%</td>
</tr>
<tr>
<td>Tokens</td>
<td>17 434</td>
<td>18 248</td>
<td>11 236</td>
<td>11 729</td>
</tr>
<tr>
<td>Wörter</td>
<td>1 558</td>
<td>1 213</td>
<td>1 196</td>
<td>1 010</td>
</tr>
<tr>
<td>einm. Wörter</td>
<td>755</td>
<td>489</td>
<td>566</td>
<td>429</td>
</tr>
<tr>
<td></td>
<td>48,46%</td>
<td>40,30%</td>
<td>47,34%</td>
<td>42,47%</td>
</tr>
</tbody>
</table>

Tabelle 4 Statistische Kenngrößen zu den Datensätzen A, B und C


Abbildung 21 Häufigkeitshistogramme für Satzlängen und Termtiefen: a) des Englischen, b) des Deutschen, c) der Dialog Acts und d) der Argumentbäume

Interessant ist ferner die Häufigkeitsverteilung der einzelnen Wortformen bzw. Symbole. In Abbildung 22 sind Ausschnitte dieser Verteilungen für deutsch, englisch sowie die Atome des Interchange Formats aufgezeigt. Die Wörter bzw. Symbole sind in je-
dem Diagramm nach ihrer Häufigkeit sortiert angegeben. Auch dieser Abbildung liegt
wieder die Menge $A$ zugrunde.

Abbildung 22 Häufigkeitshistogramme für Wort- bzw. Symbolvorkommen: a) im Englischen, b) im Deutschen, c) in den Dialog Acts und d) in den Argumentbäumen

Ein Blick auf Abbildung 23 mag eine Erklärung dafür liefern, daß die Decodierung der Wurzeln so erfolgreich ist, obwohl am Ende von Abschnitt 7.1 festgestellt wurde, daß Zuordnungen von einzelnen Wörtern des Quellsatzes zu Symbolen des Dialog Acts wegen der großen Abstraktheit seiner Bestandteile nur eingeschränkt möglich sind. Sie zeigt, daß sich ein erheblicher Teil der Wahrscheinlichkeitsmasse auf sehr wenige Konstrukte verteilt. 18

Abbildung 23 Liste der häufigsten Dialog Acts zusammen mit ihren relativen Häufigkeiten

18 Ein ähnliche Untersuchung auf älteren Datenbanken wurde von Levin et al. (2000a) durchgeführt.
Die Datensätze liegen in elektronischer Form auf der CD (vgl. S. 105) vor. Die nichttrivialen Elemente der Testmenge $C$ sind ab Seite 83 den automatisch generierten Übersetzungen gegenübergestellt.
Evaluationsdaten


Die Übersetzung erfolgte von Deutsch nach Englisch. Zu jedem deutschen Segment sind fünf englische Entsprechungen angegeben:

1. die Übersetzung eines menschlichen Experten
3. die Übersetzung des grammatischen Systems (vgl. Lavie et al., 2001)
4. die Übersetzung von JTrans, trainiert auf einem Deutsch/IF-Corpus
5. die Übersetzung von JTrans, trainiert auf Deutsch/Englisch, Englisch/IF und nach Deutsch/IF projiziert (vgl. Kapitel 5)


Die Tags der Segmente sollen das Wiederfinden in der Datenbank erleichtern.

<table>
<thead>
<tr>
<th>Segment-ID</th>
<th>Original</th>
<th>Humanübersetzung</th>
</tr>
</thead>
<tbody>
<tr>
<td>g047ak.1.3</td>
<td>hier ist Stephan Vogel.</td>
<td>this is Stephan Vogel.</td>
</tr>
<tr>
<td></td>
<td>here is Stephan Vogel.</td>
<td>Stephan's bird speaking.</td>
</tr>
<tr>
<td></td>
<td>It is.</td>
<td>This one is something.</td>
</tr>
<tr>
<td>g047ak.2.1</td>
<td>ich würde gerne Urlaub machen.</td>
<td>I would like to take a vacation.</td>
</tr>
<tr>
<td></td>
<td>I would like to take a vacation.</td>
<td>I want to travel for vacation.</td>
</tr>
<tr>
<td>g047ak.3.2</td>
<td>und zwar Skiurlaub.</td>
<td>specifically skiing vacation.</td>
</tr>
<tr>
<td></td>
<td>more precisely Skirurlaub.</td>
<td>Specifically, That.</td>
</tr>
<tr>
<td></td>
<td>Specifically, That.</td>
<td>And.</td>
</tr>
<tr>
<td>g047ak.4.1</td>
<td>im Dezember.</td>
<td>in December.</td>
</tr>
<tr>
<td></td>
<td>in December to</td>
<td>December.</td>
</tr>
<tr>
<td></td>
<td>The person December.</td>
<td>I am favorably disposed towards traveling December.</td>
</tr>
<tr>
<td>g047ak.5.2</td>
<td>ich hab' noch keinen Ort.</td>
<td>I don't have a location yet.</td>
</tr>
<tr>
<td></td>
<td>I also isn't place.</td>
<td>I would have place.</td>
</tr>
</tbody>
</table>
But I would not have another place.

Skiing is good.
It is good.

Eine Sache dabei,
daycare decide among to
I am travelling.

ich habe zwei Kinder.
I asked two children.
It would cost something.

also the kids can go pretty well too.
Where is it possible to see children’s good one also?
Can child also go skiing?
or beginners.
Or.
That.
is there accommodation with kitchen.
Küniken with accommodation?
There would be accommodation with them.
so that you can cook something yourself?
Then it is possible to cook something even also.
Also, something can?
could you also tell me,
You can tell something to me.
Please tell me.
if that is apartments with two bedrooms?
The apartments.
With children?

mit zwei Schlafzimmern.
with two Schlafzimmern.
To
That.

für Eltern, für Kinder.
for parents, for children.
It would cost something for a room.

sehr schön.
very nice.
Very nice one.

bäume im Park.
I’d like to
I want something.

vielen Dank.
thank you very much.
Thank you.

und ist das Skihaus für die Woche?
and there is a Skihaus für die Woche?
And is a week the one?

gibt es die Ermässigung für Kinder?
Would it cost something for the one for children with
discount there?

und gibt es auch Möglichkeit im Hotel oder in einer Skischule
and there is the possibility the hotel or in a ski school
child-size snowboard?
Do I rent the one at option at hotel school?

vielen Dank.
thank you very much.
Thank you.

guten Tag.
goody day tour

Not.
skiing is good.
Skiing would be good.
It is good.
one thing with that.

Because of that, is there something?
I have two children.
I have 2 children.
A child is favorably disposed towards doing something.
where kids can go pretty well too.
Where is it possible to see children’s good one also?
Can child also go skiing?
or beginners.
Or.
That.
Evaluationsdaten

Good day!

Sebastian Schluger here.

Here.

It is something.

How I connected with the tourist information?

I. The one.

I am interested in traveling with them.

Ah, very nice.

Okay.

It is very nice.

I am interested in a winter vacation near

I am interested in traveling for a winter vacation nearby.

I am interested in traveling for skiing vacation.

or in Val di Fiemme

Or at Val di Fiemme in Val di Fiemme.

or at Val di Fiemme official area.

Or is that it?

And I am looking for an accommodation.

And I am interested in an accommodation.

And I am interested in accommodation now.

specificially, I wanted to arrive on the eleventh of December.

I want to travel December the 11th.

And the person is favorably disposed towards traveling December.

Is that possible?

Is it possible to?

Something?

I see.

Okay.

I see something.

very good.

Okay.

It is something.

I already have the run

I would have the foot already.

Could I have information?

well I am mainly interested in alpine skiing.

I am interested in going skiing.

I am interested in traveling.

but wouldn't matter.

However, nothing will be available.

However, it is not.

also if I could do cross country skiing every once in a while.

When something again.

And if I can do something something.

how are the skiing areas connected by bus?

How is it the ski areas how?

Information attractions.

what about this.

How is it the one how?

What is it?

can you call the skiing area suited for families?

When it is possible to do something.

You can?

I will.

I want something.

come with my wife and two children, three and seven.

And daughter, on foot.

...
also der ältere hat schon mal einen Skikurs mitgemacht.
so the ältere Until already extend a course mitgemacht to
I am favorably disposed towards the one already.

aber die Jüngste,
but the Jüngste to
The one.

die müßte vielleicht erst,
they work maybe first to
It is possible that.

und das ist vielleicht noch ein bisschen jung,
and that is maybe have a little bit young to
And it is something.

die müßte in den Kindergartenen,
they work in the Kindergartenen.
Correct, the hotel at the hotel,

das war' natürlich sehr schön,
that would be course very nice.
It is very nice September.

wie sieht das aus mit der Verpflegung?
what about the slope with the Verpflegung?
Tell me about that one at the one.

besteht da die Möglichkeit, die Kinder dann auch
besteht that the possibility to the children Am-Fasanengarten
also
Children options there?

daß die nächtig verpflegt werden?
noticed the nächtig verpflegt will get
Then them?

meiner Frau
my wife
My wife.

ist es auch ganz lieb,
spread is it also poorly friendly to
Friendly one is somewhat nearby.

wenn sie dann mal im Hotel verweilt wird,
if they then extend the hotel verweilt will.
If it is possible to do something something,

so zwei bis
so 2 driving
That.

also mir persönlich ist das schön wichtig,
so Please persönlich is that be important.
It was important.

wir werden mit dem Auto ankommen,
we will come by car.
We are arriving by car.

und das ist ja 'n Winterurlaub ist,
and there settle uses picture vacation is,
And image is there.

besteht die Möglichkeit, das Auto in einer Garage unterzubringen?
besteht the possibility to the car in a Garage unterzubringen to
Option at the one is at the bus stop.

das ist mir schon klar.
that is Please important course.
It is already.

nur, steht das Auto dann in einer Garage
just to stehen the car Am-Fasanengarten in a Garage
well the older one has participated in a skiing course once
before.
The older one.
I want the one.

but the youngest one,
However, the most young one.
However, it is something.

she would maybe first have to,
It is possible that I must reserve the one only.
It is possible that it is the one.

and that is maybe a little young still,
Okay.
And is there?

she would need to go in the skiing kindergarten.
I require the one, the one.
It is at the hotel.

that would of course be very nice.
It is very nice.
It is very good.

what about the food?
How the food how.
Accommodation.

is there the possibility to then also the kids
The option then also.

Also, would there be attractions?
that they get fed at noon?
Then I will feed in the mid-day.
Will it be the one?

my wife.
My wife.
I am travelling.

she would also somewhat like it,
Also, it is the one.
Also, is there the one?

if she would get pampered in the hotel some time.
When you will reserve a room at the hotel.
If I am favorably disposed towards accommodation at a hotel.

about two to
Two.
That.

well for me personally that is pretty important.
Me.
It is good.

we will arrive by car.
We will arrive.
We are arriving by car.

and because this is a winter vacation,
Because it is a winter vacation.
And it is there.

is there the possibility to keep the car in a garage
The option with the car at a garage.

Is it possible to go?
I understand that.
Is that right?
It is good.

but is the car in a garage then
The car at a garage only.
The person at the one only is favorably disposed towards traveling at bus stop.

g051ak.23.2  das ist okay,
that is okay.
Hier it is.

g051ak.23.3  das ist sehr gut.
that is very good.
It is very good.

---

I am favorably disposed towards reserving a trip.

g051ak.24.3  also 'n 'a drei Sterne soll' es schon sein.
so picture picture three stars should it important fifteenth of
Three star people at image shows something already.

---

I am favorably disposed towards traveling.

---

that is alright.
It is something. Okay.
It is something.

---

That sounds good. Okay.
It is very good.
I am looking right now.
I see now.
I am favorably disposed towards traveling.
well it should at least be three stars.
I must rent.
The three star one would be good.
two stars would be fine.
Is it two star? That sounds good.
I want to stay at the three star one.

---

but that
However, the one.
But,
that would then either be that one.
Or.
If so, it is something.
that one there.
Okay.
Is there the one there?
or which one was the one before?
Or is it earlier?
Or is that it?
the Rio-Bianco would interest me.
I am interested in Rio bianco hotel.
I am interested in accommodation.
that sounds interesting.
It is interesting.
It is very interesting.
I said.
I.
I am favorably disposed towards traveling.
that would interest me.
I am interested in something.
I am interested in traveling.

---

that sounds
Is it the one?
It is something.

---

that also sounds interesting.
It is interesting.
It is interesting.
that would maybe not be bad.
It is possible that it is not the one.
It is possible that the bad one.

---

that sounds interesting.
It is interesting.
It is very interesting.
that is obviously
It is natural.
The one is something.

my wife has me here
My wife would have transportation to me here.
It is possible to go.
what would interest me first now,
what contact now first extend have actually would,
What I am interested in doing something.
I am interested in traveling.
I am interested in something now.

meine Frau, hab' ich gesehen, hat mir aufgeschrieben, ihre
my wife, asked I noticed to Until me aufgeschrieben to your
bevorzogenen Hotels.
bevorzogenen hotels.

Apprently, hotel's daughter hotel.

uns dazu entscheiden zu können,
2pm dazu entscheiden to Could to
So that you can take it.

So that you can take it.

die Skigebiete.
the ski school
It is something.

haben Sie detaillierte Informationen,
do you detaillierter information available
You would have the information.
do you have detailed information,
Would you have detailed information?

wo welche Liftanlagen sind,
where which forty-one are,
Which ski slope?
where which facility ski lifts?

bei den

to the

That.

That would be nice.
That sounds good.

It is something.

We're doing it that way then.
Yes, I do.

Because of that, we are favorably disposed towards doing something.

wie sieht das aus,
what about the slope,
Tell me about.

Tell me about.

damit ich ein Hotel noch bekommen?
damit find a hotel nothing bekomman?
Am I favorably disposed towards a hotel?

so that I will still get a hotel?

Am I favorably disposed towards a hotel?

was ich noch fragen wollte,
something else ask, there
I am doing something else called Predazzo.

I am doing something else called Predazzo.

hier und wieder nach. Ich auch gerne Eislauf,
goes and it's make i also like Eislauf,
And the person also wants to travel March.

And I also like to do ice skating.
Also, ice skating again.

And I also want to do something.

gibt es genügend Eislaufgebiete
are there genügend Eislaufgebiete
That there is.

are there enough ice skating areas

Also, is there?

Is there?

Eislaufgründe die
Eislaufgründe there

ice rinks there

Is there?

Is there?

können Sie mir vielleicht dann auch gleich noch detaillierte
können Sie mir vielleicht dann auch gleich noch detaillierte
Hotelprospekte zuschicken.
can you maybe then also coming have detaillierter Hotelpro-
spekte send.
It is possible that you also can send brochures.

It is possible to send brochures.

can you maybe then also coming have detaillierter Hotelpro-
spekte send.
can you maybe then also coming have detaillé Hotelpro-
spekte send.
It is possible that you also can send brochures.

It is possible to send brochures.

also meine Frau hat mir hier aufgeschrieben,
so my wife Until me goes aufgeschrieben to
Tell me about the one here.

well my wife wrote down for me here,

please take at your here.

that I

that I

then I

then I

I that

I that

that she would like to have something from Molina.

Do you want something?
von Hotel Latemar.
the hotel Latemar.
From the hotel from the hotel.
und dann in Cavalese hätte sie gerne vom Hotel Bellavista.
and then in Cavalese I'd they like the hotel Bellavista.
If so, hotel want accommodation from Cavalese.
das wäre sehr schön.
that would be very nice.
It is very good.
und vielleicht auch
and maybe also
And also, it is also possible that.
ja gut.
okay.
Okay.
dann gleich noch 'ne Frage speziell zum Alpin	hen coming nothing suite question particular Hotel Alpin
At downhill skiing to question specifically then.
wo wir
where feed
Is it possible?
wo ich die Kinder jetzt untergebracht habe,
where I the children now untergebracht asked to
Can you see anything at the one now?
für mich persönlich einen Tiefschnee kurs,
for me personally a Tiefschnee kurs.
And I am favorably disposed towards traveling for a child.
und wie sieht das aus mit Privat Skilehrern?
and what about the slope include Privat Skilehrern?
And who would please take on that bus.
können Sie mir da auch gleich ein Prospekt zusenden mit Preisen?
could you send me there also coming a brochure zu senden with Preisen?
Information concerning you there something containing a there.
dann glaub' ich
then think I
That.
hat' ich so
asked I so
I have it.
haben wir sowieso alles abgeklärt
do feed sowieso everything abgeklärt
We plan on everything.
von ich mich jetzt entscheid' wäre, if i contact now entschieden date should
If date is something.
kette Halbpension zu nehmen, downhill board to take,
Not.
sind das Hotel nur mit Garni,
start the hotel only with Garni so
I can start containing the hotel only.
wein man sich das individuell zusammen stellen möchte, if you somehow that individually together questions go on
And the good place is something.
also nicht in Halbpension gehen,
so not in board go to
from the hotel Latemar.
From Latemar hotel.
Accommodation from a hotel.
and then in Cavalese she would like to have one from the Hotel Bellavista.
And you want something at Cavalese from Bellavista hotel then.
I want hotel at Cavalese.
that would be very nice.
It is very nice.
It is very nice.
and probably also
And same here.
And it is also possible that.
okay.
Okay.
Okay.
then also a question on the alpine
Specifically, same question the Alpine one.
Another question.
where we
Where we.
Is it possible?
now that I have put up the kids,
I have the children now.
Can you see anything now?
for me personally a deep snow course.
Once for me.
I am interested in accommodation.
and what about private ski instructorly.
And with what payment method do I pay? Is it the one?
And accommodation.
could you also send me a brochure then with prices?
~ I F formatting error ~
Also, can you send information?
then I think
It is probably that.
I am favorably disposed towards traveling.
I have so
Would I have something?
I would have information.
we have settled everything so far
Do we have everything?
We must take.
if I should decide now,
When I am favorably disposed towards traveling.
If the person now sees something,
ot to take half-board,
I accept half board.
I want half board.
but just the hotel with breakfast.
Breakfast only.
Hotel only.
if you want to put it together individually.
When is it something, right?
If you want something.
so not to take half-board.
On foot at half board.
I want to do something at half board.
but instead then depending on that just eat on a hut here and there.

I am favorably disposed towards traveling to the food.
how much would that be in Euro?

Is that it?
then I thank you.
Thank you!
You are interested in traveling.

Not on foot.
sondern dann je nach dem auf der Hütte essen.
start then je after the on the Hütte eat.
I will do it.

Wie viel wäre das in Euro?
how much would that be in Euro?

Is that it?
dann danke ich mich bei Ihnen.
then thank you do you.
If so, you.
Weiteres zu JTrans

Die folgenden Seiten beinhalten weitere Details zum Softwaresystem, das in groben Zügen bereits in Kapitel 6 angesprochen wurde. Während dort eher inhaltliche Aspekte im Vordergrund standen, geht es hier mehr um technische Einzelheiten und Interna.

Programmstart und Kommandozeilenoptionen

Das System ist implementiert in der Sprache Java von Sun Microsystems und benötigt zur Ausführung die Version 2 (genauer JRE 1.3.1 SE) der Virtualen Maschine. Wenn $JTRANS das Verzeichnis bezeichnet, in dem sich die übersetzten Programmdateien befinden, dann läßt sich das System mit der Befehlszeile

```
java -cp $JTRANS JTrans
```

starten. Es erscheint dann ein Befehlsprompt, der zur Eingabe einer Befehlszeile auffordert.

Das Verhalten des Systems kann durch die folgenden Optionen modifiziert werden:

- `-f file` Liest die Befehle statt von der Standardeingabe aus einer Datei ein. Standardmäßig wird zudem eine Log-Datei erzeugt (s. u.). Wenn der Dateiname auf `jts` endet, kann die Endung weggelassen werden.

- `-log file` Deklariert eine Log-Datei. Wenn diese Option nicht angegeben wird und die Eingabe über den Prompt erfolgt, wird keine Log-Datei angelegt.

- `-logdir dir` Deklariert das Verzeichnis, in das die Log-Datei geschrieben werden soll. Diese Option wird nur im Zusammenhang mit `-f` beachtet.

- `-d` Schaltet den Entwicklungsmodus (debug mode) ein. Dies bewirkt eine erweiterte Fehlerausgabe beim Auftreten von Ausnahmen.

- `-c command` Führt das angegebene Kommando aus und terminiert.

Die Parameter werden in der Reihenfolge ihrer Angabe abgearbeitet. Wo sich Parameter widersprechen, gelten die Einstellungen, die von den weiter hinten stehenden vorgenommen werden. Für Parameter, die nicht erkannt werden, wird eine Warnung ausgegeben und mit der Bearbeitung fortgefahren als wäre der unbekannte Parameter nicht aufgetreten.

Der natürliche Weg zum Abbruch des Programms ist der Befehl `exit()`.
Die Befehlszeile

Die Syntax der Befehlszeile orientiert sich an der Syntax für Java-Ausdrücke. Sie erlaubt den Aufruf von Funktionen, Konstruktoren, Methoden und Zuweisungen.

Eine korrekt geformte Befehlszeile genügt der folgenden Grammatik:

\[
\begin{align*}
\langle \text{command} \rangle & \rightarrow \langle \text{expr} \rangle \mid \langle \text{def} \rangle \mid \langle \text{command} \rangle : \ast \langle \text{command} \rangle . \\
\langle \text{def} \rangle & \rightarrow \langle \text{id} \rangle := \langle \text{expr} \rangle . \\
\langle \text{expr} \rangle & \rightarrow \langle \text{id} \rangle \mid \langle \text{literal} \rangle \mid \langle \text{id} \rangle \langle \text{args} \rangle \mid \langle \text{expr} \rangle : \ast \langle \text{id} \rangle \langle \text{args} \rangle \mid \langle \text{expr} \rangle \ast \ast \langle \text{expr} \rangle . \\
\langle \text{args} \rangle & \rightarrow \langle \langle \langle \text{expr} \rangle \rangle \rangle \mid \langle \langle \langle \text{exlist} \rangle \rangle \rangle . \\
\langle \text{exlist} \rangle & \rightarrow \langle \text{expr} \rangle \mid \langle \text{expr} \rangle \ast \ast \langle \text{exlist} \rangle .
\end{align*}
\]

Ein Bezeichner (id) ist dabei jede Zeichenkette, die in Java als Bezeichner gilt. Dabei wird zusätzlich der Punkt (.) wie ein Buchstabe behandelt. Als Literale (literal) gelten 'true' und 'false' für die booleschen Werte, die Zeichenketten, die von den Konstruktoren der Klassen java.lang.Integer und java.lang.Double als int bzw. double erkannten werden, sowie String konstanten. Letztere werden wie in Java von doppelten Anführungsstrichen umgeben.

Leerzeichen zwischen Tokens werden bei der Verarbeitung einer Befehlszeile ignoriert. Das gleiche gilt für Kommentare, die wie in Java beginnend bei '// ' bis zum Zeilenumbruch oder sich von '/* ' bis zu '*/ ' erstrecken.


Zur Auswertung: Nachdem über einem eingegangen Befehl erfolgreich ein Syntaxbaum aufgebaut werden konnte, beginnt der Interpreter mit der Auswertung. Die Auswertung geschieht von links nach rechts in Tiefensuchordnung. Die Auswertung von Literalen ist trivial, zur Auswertung von Variablen führt der Interpreter eine Tabelle, die Bezeichner auf semantische Objekte abbildet. Eine Funktion bzw. ein Methodenaufufen werden ausgeführt, sobald alle nötigen Argumente ausgewertet wurden. Die Addition \( \langle \text{expr} \rangle \ast \ast \langle \text{expr} \rangle \) wird behandelt wie \( \langle \text{expr} \rangle \cdot \langle \text{expr} \rangle \cdot \). Für den Aufruf von Methoden stehen alle öffentlichen (public), nicht-statistischen Methoden zur Verfügung, die für das Objekt, auf dem sie ausgeführt werden sollen, vorhanden sind. Falls die Parameter und der Bezeichner bei einem Methodenaufufen zu mehreren Signaturen passt, wird eine Disambiguiierung nach den Regeln der Java-Spezifikation durchgeführt.
Bei der Auswertung können verschiedene Fehler auftreten: Verwendung nicht definierter Variablen, Evaluation zum null-Objekt (nicht gestattet!), Aufruf nicht definierter, nicht lizensierter oder mehrdeutiger Methoden, sowie Ausnahmen, die während der Ausführung von Methoden oder Funktionen auftreten. Das Auftreten eines dieser möglichen Fehler bewirkt den Abbruch der Auswertung an der Stelle, an der der Fehler auftrat. Es wird eine Fehlermeldung ausgegeben.

Die Implementierung des Interpreters befindet sich im Paket prompt.

**Ein- und Ausgabeströme**

Vier Datenströme, die während der gesamten Laufzeit geöffnet sind, binden das System an seine Umgebung an. Einer der Ströme liest die Befehle vom Prompt oder aus der Eingabedatei ein und verarbeitet sie wie oben erläutert. Bei den übrigen Strömen handelt es sich um Ausgabeströme, je einen für standardmäßige Ausgaben (out), Fehlernachrichten (err) sowie Protokollausgaben (log).

Beim Programmstart werden out und err durch `java.lang.System.out` und `java.lang.System.err` initialisiert, und log fügt ins Nichts bzw. in die Datei, die beim Programmbefehl als Kommandozeilenparameter angegeben wurde (s.o.).

Die Ströme lassen sich durch die Funktionen `setOut(String)`, `setErr(String)` und `setLog(String)`, die von der Standardbibliothek definiert werden, während des laufenden Systems umlenken. Über die speziellen Werte `"$out"`, `"$err"` und `"$null"` für (String) lassen sich dabei die Ströme `System.out`, `System.err` bzw. der Strom ins Leere ansprechen. Alle anderen Werte werden als Dateinamen interpretiert, bestehende Dateien werden überschrieben.

**Dateien**


Durch Implementierung der Schnittstellen `Readable` und `Writeable` läßt sich deklarieren, daß sich die Objekte einer Klasse in Binärdateien schreiben lassen. Bereitzustellen ist dabei die Konversion des Objekts in eine Folge elementarer Datentypen und umgekehrt.

Eine spezielle Textdatei ist der `Descriptor`. Durch solche Dateien werden typischerweise Objekte codiert, die einer Unterklasse von `Concept` angehören. Sie beinhalten in der ersten Zeile den Klassennamen des repräsentierten Objekts und bestehen ansonsten aus einer Liste von Zuordnungen $a = b$. Ein typischer Deskriptor für ein $n$-Gramm-
Sprachmodell ist in Abbildung 24 abgedruckt.

```
concepts.NGramLM

Gramity = 4
Lexicon = eng.lex
Partitions = 5
Probs = 14652.probs
backoff 0 = 3.344146072300438E-5
backoff 1 = 8.58000858000858E-5
backoff 2 = 1.287001287001287E-4
backoff 3 = 0.0
weight 0 = 0.7
weight 1 = 0.3
weight 2 = 0.1
```

**Abbildung 24** Deskriptor eines n-Gramm-Sprachmodells. Für Erläuterungen sei auf die Spezifikation der Klasse NGramLM verwiesen.


**Datenstrukturen**


Die Klassen legen ihre Daten in Präfixbäumen ab. Die Knoten auf der n-ten Top-Down-Ebene (vgl. S. 38) enthalten einen Index $i_n$ sowie den Wert, der dem Tupel zugeordnet wurde, das durch die Indizes auf dem Pfad von der Wurzel bis zum Knoten definiert ist. Zudem kennt jeder Knoten die Summe der Werte seiner direkten Nachfolgerknoten, wodurch eine elegante Möglichkeit zur Bestimmung von Wahrscheinlich-
keiten aus Counts (d. h. zur Normalisierung) gegeben wird. Die Knoten des Präfixbaumes verfügen über einen Zeiger auf die Wurzeln all ihrer direkten Unterbäume. Diese Zeiger werden ihrerseits in einem eigenen AVL-Baum abgelegt, so daß der Zugriff auf den Wert eines Tupels der Länge $n$ garantiert in Zeit $O((\log k)^6)$ erfolgen kann ($k$ die maximale Breite im Präfixbaum).

Wenn alle Tupel die Länge 2 haben, kann man die TupleMap als Implementierung einer dünn besetzten Matrix auffassen. Experimente zeigen, daß die Darstellung einer Matrix als TupleMap effizienter als ein zweidimensionales Array wird, sobald die Matrix zu weniger als ca. 50% gefüllt ist.

TupleMap implementiert Writeable und Readable, so daß eine TupleMap leicht in eine Binärdatei geschrieben bzw. aus einer Binärdatei gelesen werden kann.

Die Klassen TupleMap, IntTupleMap und DoubleTupleMap sind gemeinsam mit Implementierungen anderer allgemeiner Datenstrukturen im Paket util untergebracht.

**Klassenübersicht**


![Diagramm der Klassen des Pakets prompt](image_url)

**Abbildung 25** Klassen des Pakets prompt

Das Paket prompt implementiert die Verarbeitung der Befehlzeilen. Befehle werden in Form von Zeichenketten über die Methoden exec() an den Prompt übergeben, wo aus ihnen ein Expression-Baum aufgebaut wird. Wenn dies gelingt, war die Zeile syntaktisch korrekt und es wird mit der Auswertung begonnen. Dazu wird der Expression-Baum in Postfixordnung abgelaufen und dabei jedem Knoten das Value-Objekt zugeordnet, zu dem der entsprechende Unterterm evaluiert. Der Wert der Wurzel wird anschließend an den Strom out (s.o.) übergeben; wenn es sich um eine Zuweisung $y = f(x)$ handelte, werden die Variable $y$ und der Value, zu dem $f(x)$ evaluierte, in einer Tabelle von Prompt abgelegt.

Klassen, die mit Ein- und Ausgaben zu tun haben, sind im Paket io untergebracht. Hier werden die Schnittstellen Readable und Writeable deklariert, durch deren Im-
Implementierung eine Klasse als lesbar oder schreibbar deklariert wird. Ferner werden die Klassen zum Lesen und Schreiben von Binärdateien und zeilenorientierten Dateien sowie die Klasse \texttt{Descriptor} (s. o.) bereitgestellt. Nicht abgebildet ist die Klasse \texttt{OutputManager}, die die drei permanenten Ausgabestrome (s. o.) verwaltet.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image.png}
\caption{Abbildung 26 Klassen des Pakets \texttt{io}}
\end{figure}

Das Paket \texttt{util} beinhaltet eine weitgehend unzusammenhängende Sammlung häufig gebrauchter Klassen und Datenstrukturen. Hierzu gehören die \texttt{TupleMap} (s. o.) sowie verschiedene Implementierungen von Iteratoren, Mengen, Multimengen, Warteschlangen usw.

Die Pakete \texttt{prompt}, \texttt{io} und \texttt{util} stellen eine allgemeine Plattform zur Verfügung, die unabhängig von Rest auch in anderen Situationen anwendbar sein sollte. Alle für die Sprachübersetzung relevanten Klassen befinden sich in den Paketen \texttt{base} und \texttt{concepts}, die Klassen zur Behandlung von Termen im Paket \texttt{nespolo}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image2.png}
\caption{Abbildung 27 Klassen des Pakets \texttt{base}}
\end{figure}
Weiteres zu JTrans


Abbildung 28 Klassen des Pakets concepts

Annahmen über Datenformate im Nespole!-Projekt

Zur Generierung von IFPhrase-Objekten aus Zeichenketten bedient sich ein IFLexicon eines Parsers, der gemeinsam von den Klassen Root und Tree bereitgestellt wird.
Diesem Parser liegt die folgende kontextfreie Grammatik zugrunde:

\[
\begin{align*}
\langle \text{top} \rangle & \rightarrow \langle \text{root} \rangle \mid \langle \text{root} \rangle \langle \text{tree} \rangle, \\
\langle \text{root} \rangle & \rightarrow \langle \text{id} \rangle \mid \langle \text{id} \rangle \langle \text{concepts} \rangle, \\
\langle \text{concepts} \rangle & \rightarrow \langle \text{id} \rangle \mid \langle \text{concepts} \rangle \mid \langle \text{id} \rangle, \\
\langle \text{tree} \rangle & \rightarrow \langle \langle \text{args} \rangle \rangle, \\
\langle \text{args} \rangle & \rightarrow \langle \langle \text{id} \rangle \rangle \mid \langle \langle \text{id} \rangle \rangle \langle \text{val} \rangle \mid \langle \langle \text{id} \rangle \rangle \langle \text{val} \rangle \mid \langle \langle \text{val} \rangle \rangle \langle \langle \text{args} \rangle \rangle, \\
\langle \text{val} \rangle & \rightarrow \langle \langle \text{id} \rangle \rangle \mid \langle \langle \text{tree} \rangle \rangle \mid \langle \langle \text{set} \rangle \rangle, \\
\langle \text{set} \rangle & \rightarrow \langle \langle \langle \text{vals} \rangle \rangle \rangle \mid \langle \langle \langle \text{coord} \rangle \rangle \rangle, \\
\langle \text{vals} \rangle & \rightarrow \langle \langle \text{val} \rangle \rangle \mid \langle \langle \text{val} \rangle \rangle \langle \langle \text{vals} \rangle \rangle, \\
\langle \text{coord} \rangle & \rightarrow \langle \langle \text{val} \rangle \rangle \mid \langle \langle \text{val} \rangle \rangle \langle \langle \text{val} \rangle \rangle \langle \langle \text{vals} \rangle \rangle.
\end{align*}
\]

Gültige Bezeichner \( \langle \text{id} \rangle \) sind Zeichenketten, die keine Leerzeichen und keines der Zeichen `;` `+` `&` `;` `=` `(` `)` `;` `;` `;` enthalten. Groß- und Kleinschreibung wird nicht unterschieden. Leerzeichen zwischen Terminalsymbolen werden überlesen. Die \textit{Coordination} \( \langle \text{coord} \rangle \) ist nicht mehr Teil des Interchange Formats, wird aber aus historischen Gründen noch weiter berücksichtigt. Semantisch wird \( \langle \langle \text{val} \rangle \& \ldots \rangle \) wie \( \langle \langle \text{val} \rangle \ldots \rangle \) behandelt.


Zur Konstruktion eines \textit{IFLexicon} sind drei Dateien nötig, die die \textit{Spezifikation} des Interchange Formats beinhalten. Eine Datei \texttt{da.db.lsp} definiert die Atome des Dialog Acts (der Wurzel), eine weitere mit Namen \texttt{nespole-arguments.db.lsp} die Argumente und eine dritte mit Namen \texttt{nespole-values.db.lsp} Werte und Wertklassen (IF, 2002).

Die Datei \texttt{da.db.lsp} beinhaltet eine Folge von Definitionen. Jede dieser Definition definiert genau einen Speech Act oder ein Konzept. Dazu wird eine Liste möglicher Fortsetzungen (engl. continuations) sowie eine Liste von Argumenten angegeben, die vom definierten Atom lizensiert werden. Eine Fortsetzung kann mehrere Symbole umfassen. Die Spezifikationsdateien verwenden die Syntax von Lisp, die Definition eines Speech Acts oder Konzepts hat dabei die Form

\[
\begin{align*}
\langle \text{def} \rangle & \rightarrow \langle \langle \langle \langle \text{id} \rangle \langle \text{conts} \rangle \langle \text{args} \rangle \rangle \rangle \rangle, \\
\langle \text{conts} \rangle & \rightarrow \langle \text{nil} \rangle \mid \langle \langle \langle \text{cont} \rangle \rangle \rangle, \\
\langle \text{cont} \rangle & \rightarrow \langle \langle \langle \text{id} \rangle \rangle \rangle \mid \langle \langle \langle \text{id} \rangle \rangle \langle \text{val} \rangle \rangle \mid \langle \langle \langle \text{id} \rangle \rangle \langle \text{val} \rangle \rangle \langle \langle \text{args} \rangle \rangle, \\
\langle \text{args} \rangle & \rightarrow \langle \langle \langle \text{id} \rangle \rangle \rangle \langle \langle \text{val} \rangle \rangle \langle \langle \text{vals} \rangle \rangle.
\end{align*}
\]

Aufeinanderfolgende Bezeichner sind durch Leerzeichen voneinander zu trennen.
In `nespole-arguments.db.lsp` befinden sich die Definitionen der Argumente. Eine Argumentdefinition besteht aus mehreren Listen, jede dieser Listen ist überschrieben mit `:ISA`, `:VALUES` (Definition der Köpfe), `:ATTRIBUTES` (Definition der Attributargumente), `:RELATIONS` (Definition der Relationsargumente), `:COMMENTS` (Kommentare) oder `:GOTO` (Import einer anderen Argumentdefinition). Zu einem Argumentbezeichner kann es mehrere unabhängige Definitionen geben. Syntaktisch wird für eine Argumentdefinition das Format

\[
\langle \text{arg} \rangle \rightarrow (\langle \text{id} \rangle (\text{def})^+)'.
\]

\[
\langle \text{def} \rangle \rightarrow (\langle \text{list} \rangle^+)'.
\]

\[
\langle \text{list} \rangle \rightarrow (\langle \text{slot} \rangle (\langle \text{id} \rangle^+)'.
\]

\[
\langle \text{slot} \rangle \rightarrow :ISA | :VALUES | :ATTRIBUTES
\]

\[
| :RELATIONS | :COMMENTS | :GOTO'.
\]


Die Menge aller atomaren Werte ist gegeben durch all jene Werte, die von irgendeinem Argument lizensiert werden (s.o.). `nespole-values.db.lsp` dient also nur zur Definition der Werteklassen. Eine Werteklassendefinition \( c := (w_1, \ldots, w_n) \) wird syntaktisch dargestellt in der Form \( (c, w_1, \ldots, w_n) \). Die \( w_i \) können dabei ihrerseits wieder Klassen von Werten sein. Neben den auf diese Weise definierten endlichen Klassen gibt es die drei unendlichen Klassen \([\text{untranslated-string}]\) (für nicht übersetzende Texte, umgeben von `"`) \([n]\) (für Zahlen) und \([\text{nth}]\) (für Ordnungszahlen).

Die unendlichen Klassen werden offiziell in `nespole-arguments.db.lsp` definiert, indem in einem Kommentar ihre Elemente beschrieben sind. In `IFlexicon` sind sie jedoch fest eingecodiert, die Definitionen in der Spezifikationsdatei sind auszukomentieren.
Iterationsalgorithmen

Das effiziente Aufzählen von Tupeln \((t_1, \ldots, t_m) \in \{1, \ldots, k\}^m\) mit gewissen Eigenschaften erfordert Algorithmen, die einerseits zu kompliziert sind, als daß man sie direkt implementieren könnte, die aber andererseits offenbar nicht kompliziert genug sind, als daß man sie veröffentlichen würde. Das einzige dem Autor bekannte Werk, das sich diesen Algorithmen widmet, ist Knuth (2002), doch den gesuchten Algorithmus findet man auch dort nicht.

Ein effizienter Iterationsalgorithmus ist im Baumdecoder (Abschnitt 4.4 auf Seite 43) nötig, um alle Mengen von Hypothesen aufzuzählen, die zu einer neuen Hypothese ergänzt werden sollen. Allgemeiner gesprochen ist also ein Algorithmus gesucht, der eine minimale Menge von \(m\)-Tupeln \((t_1, \ldots, t_m) \in \{1, \ldots, k\}^m\) aufzählt, aus denen durch Permutation alle \(m\)-Tupel generierbar sind, für die \(t_i \neq t_j\) (\(i \neq j\)) gilt. Für \(k = 5\) und \(m = 3\) werden die Tupel \((1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4), (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 5), (3, 4, 5)\). Diese Tupel geben genau die Möglichkeiten an, \(m\) Kugeln ohne Zurücklegen aus einer Urne mit \(k\) Kugeln zu ziehen, und dabei die Reihenfolge außer Acht zu lassen. Dies entspricht den Einschränkungen des Exchange Formats, nach denen jedes Unterargument höchstens einmal pro Rahmen auftreten darf und die Reihenfolge der Unterterme innerhalb eines Rahmens ohne Bedeutung ist.

Bekanntlich gibt es genau \(\binom{k}{m} = \frac{k!}{(k-m)! m!}\) solche Möglichkeiten, und dieser Anhang wird in drei Schritten einen optimalen Algorithmus zur Aufzählung dieser Möglichkeiten entwickeln.

Alle Tupel

Alle \(m\)-Tupel ohne Einschränkung optimal aufzuzählen ist noch kein Problem. Es handelt sich dabei gerade um die iterierte Inkrementierung der \(k\)-näheren Zahlen. Algorithmus 3 zeigt eine Lösung. Die Notation des Algorithmus verwendet das Entwurfs muster der Template Method (vgl. Balzert, 2001; Knuth, 2002): Der Aufruf

\[
\text{work with the tuple}(t_1, \ldots, t_m)
\]

in Zeile 4 markiert die Stelle, an der das Tupel \((t_1, \ldots, t_m)\) verwendet werden soll.

Mit Hilfe einer vollständigen Induktion sieht man leicht die Korrektheit von Algorithmus 3 ein. Für \(m = 1\) ist die for-Schleife leer, und der Algorithmus zählt bloß die Zahlen von 1 bis \(k\) ab. Nach Induktionsvoraussetzung sei der Algorithmus korrekt für \(m - 1\), d. h. alle \((m - 1)\)-Tupel \((t_2, \ldots, t_m)\) werden korrekt aufgezählt. Dann wächst im Induktionsschritt die Länge der for-Schleife um einen Schritt, und dieser zusätzliche Schritt bewirkt die Reinitialisierung des übergelaufenen \((m - 1)\)-Tupels sowie die Inkrementierung von \(t_1\), \(t_1\) war zu 1 initialisiert und läuft bis \(k\) (bei \(t_1 > k\) terminiert die äußere Schleife). Für jeden Wert \(t_1 \in \{1, \ldots, k\}\) werden nach Induktionsvoraussetzung
alle \((m-1)\)-Tupel \((t_2, \ldots, t_m)\) aufgezählt, insgesamt also alle \(m\)-Tupel \((t_1, \ldots, t_m)\). Dies zeigt die Korrektheit.

Algorithmus 3 Aufzählen aller \(m\)-Tupel \((t_1, \ldots, t_m) \in \{1, \ldots, k\}^m\)

Die Zeitkomplexität dieses Algorithmus beträgt \(O(k^m)\) Schritte: Für jedes der \(k^m\) Tupel
ist ein Schritt (Zuweisung und/oder Addition) auf \(t_m\) nötig. Das Element \(t_{m-1}\) wird
dagegen nur einmal pro \(k\) Tupel verändert und liefert somit einen Beitrag von \(k^{m-1}\)
Operationen. Allgemein wird das Element \(t_{m-i}\) genau \(k^{m-i}\) mal verändert, jeweils mit
einer konstanten Anzahl von Operationen. Damit ergibt sich

\[
x(m, k) = \sum_{i=0}^{m-1} k^i = \frac{k^{m+1} - k}{k - 1} = O(k^m).
\]

Da der Algorithmus zur Aufzählung der \(k^m\) Tupel jedes Tupel einmal anfassen muß, ist
dies auch die untere Schranke für die Komplexität des Problems. Er ist somit optimal.

Alle Tupel modulo Permutation

Wenn man von all jenen Tupeln, die sich durch Permutation ineinander überführen
lassen, je nur eines generieren will, bietet sich die Beschränkung auf eine kanonische
Form an. So kann man sich z. B. auf all jene Tupel \((t_1, \ldots, t_m)\) mit \(t_i \leq t_j \ (i < j)\)
beschränken. Algorithmus 3 ließe sich trivial adaptieren, indem man die ungewollten
Tupel herausfiltert, somit also Zeile 4 durch

4A \[ \textbf{if } \forall i < j : t_i \leq t_j \text{ then} \]
4B \[ \text{work with the tuple} (t_1, \ldots, t_m); \]

ersetzte. Der resultierende Algorithmus behielte dann jedoch die gleiche Komplexität
\(O(k^m)\) und wäre nicht optimal, da man bloß an \(O(k^m)\) Tupeln interessiert ist.

Eine bessere Lösung zeigt Algorithmus 4. Hier werden bei Überlauf eines Untertupels
\((t_{i+1}, \ldots, t_m)\) nach Inkrementierung von \(t_i\) die \(t_j\) \((j = i+1, \ldots, m)\) nicht wie oben zu \(1\)
initialisiert sondern zu \(t_i\) reinitialisiert. Auf diese Weise zählt der Algorithmus genau
die Tupel mit monoton steigenden Indizes auf. Aus der Tatsache, daß jedes solche
Tupel kanonischer Repräsentant einer Permutationsklasse ist, folgt die Korrektheit des Algorithmus.

Gegenüber Algorithmus 3 führt Algorithmus 4 keine zusätzlichen Schritte aus. Die zusätzliche for-Schleife in den Zeilen 10 und 11 führt genau die Zuweisungen aus, die in Algorithmus 3 in Zeile 9 stehen. Da es $m!$ Permutationen gibt und jedes gelieferte Tupel Repräsentant einer Klasse von $m!$ Tupeln ist, erhält man die Komplexität $\zeta(m,k) = O\left(\frac{k^m}{m!}\right)$. Dies ist wieder ein optimales Ergebnis.

```
procedure tuple iterator2(m, k)
  (t₁, ..., tₘ) := (1, ..., 1);
  while t₁ ≤ k do
    work with the tuple(t₁, ..., tₘ);
    tₘ := tₘ + 1;
  for i = m downto 2 do
    if tᵢ > k then
      tᵢ₋₁ := tᵢ₋₁ + 1;
    else
      for j = i+1 to m do
        tᵢ := tᵢ;
      exit for
```

**Algorithmus 4** Aufzählen aller $m$-Tupel $(t₁, ..., tₘ) \in \{1, ..., k\}^m$ modulo Permutation

**Alle Möglichkeiten, Lotto zu spielen**

Gesucht sind nun alle $m$-Tupel $(t₁, ..., tₘ)$ mit $tᵢ \neq t_j$ ($i \neq j$), von denen sich keine zwei durch Permutation eineinander überführen lassen. Dies ist gerade die Situation beim Lotto 6 aus 49: Dort wählt man $m = 6$ verschiedene Zahlen aus $k = 49$ und berücksichtigt deren Reihenfolge nicht.

Die Möglichkeiten, $m$ Kugeln ohne Zurücklegen aus einer Urne mit $k$ Kugeln zu ziehen, werden beschrieben durch die Tupel $(t₁, ..., tₘ)$ mit $tᵢ \neq t_j$ ($i \neq j$). Es gibt genau $k(k-1)(k-2)\cdots(k-m+1)$ solche Tupel. Nach Ziehen der $i$-ten Kugel bleiben in der Urne noch $k-i$ Kugeln zurück. Bei einem optimalen Iterationsalgorithmus wird daher $tᵢ$ nur über $k-i$ Werte iterieren.

Da zudem die Reihenfolge innerhalb der Tupel nicht interessiert (d. h. es soll für jede Reihenfolge nur ein Tupel erzeugt werden), kann man Algorithmus 4 modifizieren. Dort wurden durch Verwendung eines Monotoniekriteriums alle $m$-Tupel modulo Permutation aufgezählt. Wenn man dort die Monotonie durch eine strenge Monotonie ersetzte, werden zusätzlich zu den nicht monotonen Tupeln auch jene übersprungen, die mindestens zwei übereinstimmende Elemente haben. Dies führt auf Algorithmus 5.

Pro Tupel hat sich die Komplexität dieses Algorithmus gegenüber der von Algorithmus 4 nicht verändert. Beidesmal ist die amortisierte Komplexität (vgl. Cormen et al., 2001, Kapitel 17) konstant und damit optimal. Da hier nur noch $\binom{k}{m}$ Tupel
aufgezählt werden, benötigt der Algorithmus demnach also \( O\left(\binom{k}{m}\right) \) Schritte.

1 \textbf{procedure} tuple iterator3\((m,k)\)
2 \((t_1, \ldots, t_m) := (1, \ldots, m)\);
3 \textbf{while} \( t_i < k - m \) \textbf{do}
4 \quad \text{work with the tuple}(t_1, \ldots, t_m);
5 \quad t_m := t_m + 1;
6 \quad \textbf{for} \( i = m \) \textbf{downto} 2 \textbf{do}
7 \quad \quad \textbf{if} \( t_i > k \) \textbf{then}
8 \quad \quad \quad t_{i-1} := t_{i-1} + 1;
9 \quad \quad \textbf{else}
10 \quad \quad \quad \textbf{for} \( j = i + 1 \) \textbf{to} m \textbf{do}
11 \quad \quad \quad \quad t_j := t_i + (j - i);
12 \quad \quad \textbf{exit} \textbf{for}

\textbf{Algorithmus 5} Aufzählung aller Möglichkeiten, \( m \) Kugeln mit Zurücklegen ohne Beachtung der Reihenfolge aus einer Urne mit \( k \) Kugeln zu ziehen.
Begleitmaterial

Teil dieser Arbeit ist eine CD-ROM, auf der Begleitmaterial zusammengestellt ist. Neben der verwendeten Literatur sind darin auch das entwickelte Softwaresystem inkl. Dokumentation und Quellen sowie die verwendeten Trainings- und Testmengen enthalten.

Es folgt eine Übersicht über die Materialien auf der CD, nähere Informationen sind auf der CD selbst zu finden.

/bibliography  Sammlung der verwendeten Literatur
/corpora      Sammlung verschiedener Corpora
/deNews       Bilinguales Corpus Deutsch/Englisch
/hkNews       Bilingualer Corpora Deutsch/Chinesisch
/nespole      Datenbanken von Nespole! in verschiedenen Format und Versionen
/verbmobil    Bilinguales Corpus Deutsch/Englisch aus dem Verbmobil-Projekt samt Testmenge
/evaluation   Evaluationsdaten: Hypothesen und Statistiken
/alignments   Daten zu Abschnitt 7.1
_corpusSize   Daten zu Abschnitt 7.5 (Extrapolation)
/human        Daten zu Abschnitt 7.3
/if           Daten zu Abschnitt 7.2
/overtraining Daten zu Abschnitt 7.5 (Übertrainieren)
/if-spec      Spezifikationsdateien für das Interchange Format in verschiedenen Versionen
/v2001.11     Version vom November 2001, passend zur Trainingsmenge B
/v2002.4      Version vom April 2002, passend zur Trainingsmenge A sowie zu den Referenzen der Testmenge
/paper        Das Papier „Interlingua Based Statistical Machine Translation“, in dem die Ergebnisse dieser Arbeit in der erforderlichen Kürze dargestellt werden.
/software     Software
/trans        Das auf den Seiten 53–58 und 91–99 beschriebene Softwaresystem
/class        Ausführbarer Java-Bytecode
/doc          Vollständige Dokumentation aller Pakete, Klassen und Methoden
/html         HTML-Version des Quellcodes
/sessions     Sammlung von Beispielsitzungen
<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/src</td>
<td>Quellcode</td>
</tr>
<tr>
<td>/talk</td>
<td>Folien zum Diplomarbeitsvortrag</td>
</tr>
<tr>
<td>/thesis</td>
<td>Die vorliegende Ausarbeitung</td>
</tr>
</tbody>
</table>
Symbolverzeichnis

$O(f(n))$ Komplexitätsbetrachtung im O-Kalkül
$p(A)$ Wahrscheinlichkeit für das Ereignis $A$
$p(A|B)$ Bedingte Wahrscheinlichkeit für das Ereignis $A$ unter der Bedingung $B$

$e$ eulersche Zahl
$log x$ natürlicher Logarithmus von $x$
$e = (e_1, \ldots, e_i, \ldots, e_j)$ Wortfolge $e$, bestehend aus den $i$ Wörtern $e_1$ bis $e_j$
$l = |e|$ Länge des Satzes $e$
$f = (f_1, \ldots, f_j, \ldots, f_m)$ Wortfolge $f$, bestehend aus den $m$ Wörtern $f_1$ bis $f_m$
$m = |f|$ Länge des Satzes $f$
$g = (g_1, \ldots, g_k, \ldots, g_n)$ Wortfolge $g$, bestehend aus den $m$ Wörtern $g_1$ bis $g_n$
$n = |g|$ Länge des Satzes $g$
$a = (a_l, \ldots, a_m)$ Alignment, Funktion, die jeder Quellsatzposition $j$ eine Zielsatzposition $a_j$ zuordnet.

$t(f|e)$ Übersetzungswoahrscheinlichkeit, Wahrscheinlichkeit, daß $f$ die Übersetzung des Wortes $e$ ist
$\delta(x,y)$ Kroneckersymbol, 1, falls $x = y$ und 0 sonst.
$c(f|e, f, e)$ Counts für die Übersetzungswoahrscheinlichkeiten den IBM Modellen
$a(i, j, m, l)$ Zuordnungswahrscheinlichkeit, Wahrscheinlichkeit dafür, daß $a_j = i$ gilt
$c(i, j, m, l, e, f)$ Counts für die Zuordnungswahrscheinlichkeiten im IBM Modell 2

$n(\varphi, e_j)$ Fertility-Wahrscheinlichkeit im IBM-3-Modell
$P(T)$ Perplexität einer Menge $T$
$PP(T)$ Phrasenperplexität der Menge $T$
$\delta t$ Tiefe des Terms $t$
$\delta_1(s,t)$ Bottom-Up-Tiefe des Symbols $s$ im Term $t$
$\delta_1(s,t)$ Top-Down-Tiefe des Symbols $s$ im Term $t$
$L_s(i, t)$ Bottom-Up-Ebene $i$ im Term $t$
$L_s(i, t)$ Top-Down-Ebene $i$ im Term $t$
$top_s(t)$ $i$-ter Kopf des Terms $t$
$|A|$ Mächtigkeit der Menge $A$
$\text{wer}(t_1, t_2)$ Wortfehlerrate eines Baums $t_1$ bezgl. $t_2$
$[a, b)$ Intervall, Menge aller reellen Zahlen $x$ mit $a \leq x \leq b$
$[a, b]$ halboffenes Intervall, Menge aller reellen Zahlen $x$ mit $a \leq x < b$
$\mathbb{R}$ Menge aller reellen Zahlen
$(w)$ Wortklasse $w$
$A(f)$  Abbildung des Wortes $f$ in seine Quellsprach-Wortklasse
$B(e)$  Abbildung des Wortes $e$ in seine Zielsprach-Wortklasse
$\omega(n)$ Komplexität eines Algorithmus in Abhängigkeit von $n$
Literatur


### Index

**A**
- a *(Method)* .................................. 55
- Abstand ...................................... 20, 50
- Adapter ........................................ 57
- add *(Method)* ................................ 55
- Agent ........................................... 32
- Aktion ✧ *Speech Act* 
- Alignment ....................................... 24, 107
- Alignment Probability 
  ✧ Zuordnungswahrscheinlichkeit 
- Annotation ..................................... 49, 75
- Äquivalenzklasse .............................. 41
- Argument ....................................... 16, 32, 99
  ✧ Attribut ....................................... 33, 99
  ✧ -baum ......................................... 47
  ✧ Hilfs- .......................................... 46
  ✧ Kopf ........................................... 33
  ✧ Relations ...................................... 33, 99
- Argument *(Klasse)* .......................... 57
- Assymptote .................................... 73
- Atom ✧ *Symbol od. Wortform* 
- aufzählen ....................................... 101–104
- Auswertung ..................................... 92

**B**
- Backing Off ................................... 22, 28, 40
- Baum ✧ *Term* 
  ✧ Präfix ......................................... 94–95
- Baumsprache ✧ *Termsprache* 
- Bayes, Thomas, Regel von ..................... 20
- bedingte Wahrscheinlichkeit .................. 107
- Befehlsprompt ................................... 91
- Befehlszeile .................................... 92–93
- Bertram, Benjamin ............................. 18
- Bezeichner ...................................... 92
- Bibliothek ...................................... 53
- Std ............................................... 53
- Bigramm ........................................ 22
- Blatt ............................................ 37
- boolean *(Datentyp)* ......................... 53

**C**
- Bottom-Up-Ebene ............................. 38
- Bottom-Up-Tiefe .............................. 38
- Breite .......................................... 37
- Carnegie Mellon University ............... 32
- CD .............................................. 105–106
- Chapman, Sydney ............................. 50
- Client .......................................... 32
- Concept ✧ *Konzept* 
- Concept *(Klasse)* ........................... 53, 54, 57, 93
- Coordination ................................... 98
- Corpus ......................................... 15, 19, 20, 49, 79–82
  ✧ Addition ....................................... 55
  ✧ Dimension ..................................... 54
  ✧ Konkatenation ................................ 55
  ✧ Multiplikation ................................. 55
- Corpus *(Klasse)* ............................... 54–56, 58, 97
- Count .......................................... 25, 27, 42, 94, 107
- Customer ....................................... 32

**D**
- Datentyp ........................................ 53
- debug mode ..................................... 91
- Decoder *(Klasse)* ............................ 54, 55, 57
- Decodierung ✧ *Decoding* 
- Decoding ......................................... 20, 28–30, 43–46, 57, 75, 76
- Descriptor *(Klasse)* ......................... 93, 94, 96
- Dialog .......................................... 79
- Dialog Act ..................................... 32
  ✧ siehe auch ✧ *Wurzel* 
- Dimension ..................................... 54
- Disambiguierung ...................... .......................... 66
- Diskursbereich ................................. 32
- Divergenz, strukturelle ..................... 50
- Dolmetscher ................................... 31
- Domain Action ................................. 32
  ✧ double *(Datentyp)* ......................... 53
- DoubleTupleMap *(Klasse)* .................. 94, 95
- Dreiecksungleichung ......................... 50
Index

E
Ebene ........................................... 38, 107
Bottom-Up ..................................... 38
Top-Down ...................................... 38
Editierdistanz ................................. 72
end of sentence ............................... 30
End-to-End-Evaluation ...................... 66
Entwicklungsumodus ........................ 91
Evaluation ................................... 59–73, 83–91
exec (Methode) .............................. 95
exit (Funktion) ............................... 53, 58
Expression (Klasse) ......................... 95
Extrapolation ................................. 72

F
Fehler ............................................ 62
Fertility ....................................... 28, 76, 107
(first-name) ................................. Wörkasse, (first-name)
Flexibilität .................................... 15, 75
Frame .......................................... 33
Fügen, Christian ............................. 18
Füllwort ....................................... 60
Funktion ...................................... 53, 92
Funktionssymbol ............................ Symbol

G
Gates, Donna ................................... 18
Generierung ................................... 15, 16, 60
getFilename (Methode) ..................... 94
getPhraseIterator (Methode) ............. 55
getStringIterator (Methode) ............. 55
Gewicht ....................................... Interpolation
Goethe, Johann Wolfgang von ............ 15
goto ........................................... 99
Grammatik ..................................... 35–36, 77

H
Head ............................................. Kopf
Hilfsverteilung ................................. 51
Hotelsreservierung ......................... 32
Hypothese ................................... 29, 43
Bewertung ................................... 29, 44

I
IBM-Modelle ................................. 24–28, 49, 55, 75
für Bäume .................................. 42
höhere ....................................... 28, 75–76
vereinfachte .................................. 43
Ibm1TM (Klasse) ....................... 54–56, 58
Ibm2TM (Klasse) ....................... 54–56, 58
IFDecoder (Klasse) ....................... 57
IFElement (Klasse) ....................... 56, 57
IFLexicon (Klasse) ....................... 56–58, 97–99
IFLM (Klasse) .............................. 57, 58
IFPhrase (Klasse) ......................... 56, 57, 97, 98
IFTM (Klasse) .............................. 57, 58
Implementierung ........................ JTrans
Indexmenge ................................... 41
Indizierung .................................. 41
Informatik ................................... 15
int2string (Methode) ...................... 54
int (Datentyp) .............................. 53
Interchange Format ....................... 17, 32–34, 37, 44, 46–47
Gratifikat ................................... 98
Normalform .................................. 39
Parser ......................................... 97
Spezifikation ................................ 98
Spezifikationsdateien ..................... 105
Interlingua ................................ 15, 16, 32, 75
Interpolation ............................... 22, 28, 40
Intervall ...................................... 107
IntReader (Klasse) ......................... 93
IntTupleMap (Klasse) .................... 94, 95
IntWriter (Klasse) ......................... 93
isa ........................................... 99
Iterator ..................................... 55, 101–104

J
Java ............................................. 91
JTrans ......................................... 53–58, 83, 91–99
a (Methode) ................................. 55
add (Methode) .............................. 55
Argument (Klasse) ....................... 57
Auswertung ................................... 92
Befehlsprompt ............................... 91
Bedeutungszeile ............................. 92–93
Beispiel ..................................... 53, 56, 58
Bezeichner .................................. 92
Bibliothek .................................. 53
boolean (Datentyp) ....................... 53
Concept (Klasse) ......................... 53, 54, 57, 93
Corpus (Klasse) .......................... 54–56, 58, 97
Datentyp .................................... 53
debug mode ................................. 91
Decoder (Klasse) ......................... 54, 55, 57
Index

Descriptor (Klasse) ...................... 93, 94, 96

double (Datentyp) ..................... 53
DoubleTupleMap (Klasse) .......... 94, 95
Entwicklungsmodus .................... 91
e (Klasse) ................................ 95
exit (Funktion) .......................... 53, 58
Expression (Klasse) .................. 95
Funktion .................................. 53, 92
getFilename (Methoden) ............. 94
getPhraseIterator (Methoden) ...... 55
getStringIterator (Methoden) ...... 55
Ibm1TM (Klasse) .......................... 54–56, 58
Ibm2TM (Klasse) .......................... 54–56, 58
IFDecoder (Klasse) ..................... 57
IFElement (Klasse) ...................... 56, 57
IFLexicon (Klasse) ..................... 56–58, 97–99
IFLM (Klasse) ............................. 57, 58
IFPhrase (Klasse) ........................ 56, 57, 97, 98
IFTM (Klasse) ............................. 57, 58
int2string (Methoden) ............... 54
int (Datentyp) ........................... 53
IntReader (Klasse) ..................... 93
IntTupleMap (Klasse) ................. 94, 95
IntWriter (Klasse) ..................... 93
Kommentare .............................. 92
Konstruktor .............................. 92
LanguageModel (Klasse) .............. 54, 55, 57, 94
learn (Methoden) ...................... 55, 56, 58
Leerzeichen .............................. 92
Lexicon (Klasse) ....................... 54–57, 94, 97
LineReader (Klasse) ................... 93
LineWriter (Klasse) .................... 93
Literale .................................. 92
load (Funktion) ........................... 53
Log-Datei ................................. 91
long (Datentyp) .......................... 53
Methode .................................. 53, 54, 92
m (Klasse) ................................ 53
macht (Methoden) ....................... 55
Navigationssymbol .................... 93
new .......................................... 92
NgramLM (Klasse) ....................... 54–56, 58, 94
OutputManager (Klasse) .............. 96
Phrase (Klasse) .......................... 54–57, 97
phrase2string (Methoden) .......... 54
PhraseFactory (Klasse) ............... 97
PhraseIterator (Klasse) .............. 55, 97
PhraseIteratorAdapter (Klasse) .... 97
Prompt (Klasse) .......................... 95
Quellcode ................................. 105, 106
read (Funktion) .......................... 53
Readable (Klasse) ....................... 93, 95
Root (Klasse) ............................. 56, 57, 97
RootDecoder (Klasse) ................. 57
RootIbm2Decoder (Klasse) ............ 58
RootLM (Klasse) .......................... 57, 58
RootTM (Klasse) .......................... 57, 58
saveTo (Methoden) ..................... 94
sctDecoder (Methoden) ............... 56, 58
setErr (Funktion) ....................... 93
setLexicon (Methoden) .............. 55, 58
sctLog (Funktion) ...................... 93
setOut (Funktion) ...................... 93
SpeechActPart (Klasse) .............. 87
Std ........................................... 53
string2int (Methoden) ............... 54
string2phrase (Methoden) .......... 54
String (Datentyp) ...................... 53
StringIterator (Klasse) .............. 55, 97
StringIteratorAdapter (Klasse) ...... 97
Syntax ..................................... 92–93
t (Methoden) .............................. 55
time (Funktion) ........................... 53
trans (Methoden) ....................... 55, 56, 58
TranslationModel (Klasse) .......... 54, 55, 57, 97
Tree (Klasse) ............................. 56, 57, 97
TreeDecoder (Klasse) ................. 57
TreeIbm1TM (Klasse) ................. 57, 58
TreeIbm2Decoder (Klasse) .......... 57, 58
TreeIbm2TM (Klasse) ................. 57, 58
TreeLM (Klasse) .......................... 57, 58
TreeSlbm2TM (Klasse) ............... 57
TreeTbm2TM (Klasse) ................. 58
TreeTM (Klasse) .......................... 57
TupelMap (Klasse) ...................... 94–96
type (Funktion) ........................... 53
Typumwandlung ......................... 53
updateDescriptor (Methoden) ...... 94
Value (Klasse) ........................... 57, 95
Writeable (Klasse) ..................... 93, 95
WWDecoder (Klasse) ................... 54–56
Zuweisung ................................. 92

K
Kanal, verrauscht ...................... 19–20
<table>
<thead>
<tr>
<th>Index</th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projektion</td>
<td>17, 49–52, 55, 69–71, 75</td>
</tr>
<tr>
<td>Glättung</td>
<td>49</td>
</tr>
<tr>
<td>Hilfsverteilung</td>
<td>51</td>
</tr>
<tr>
<td>Verlust</td>
<td>49–50</td>
</tr>
<tr>
<td>Prompt (Klasse)</td>
<td>95</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>Quellcode</td>
<td>105, 106</td>
</tr>
<tr>
<td>Quellsatz</td>
<td>19</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Rahmen</td>
<td>33</td>
</tr>
<tr>
<td>Raum, metrischer</td>
<td>50</td>
</tr>
<tr>
<td>Rauschen</td>
<td>21, 49, 75</td>
</tr>
<tr>
<td>read (Funktion)</td>
<td>53</td>
</tr>
<tr>
<td>Readable (Klasse)</td>
<td>93, 95</td>
</tr>
<tr>
<td>Regelsystem</td>
<td>75</td>
</tr>
<tr>
<td>Reichert, Jürgen</td>
<td>18</td>
</tr>
<tr>
<td>Reiseplanung</td>
<td>32</td>
</tr>
<tr>
<td>Robustheit</td>
<td>15, 75</td>
</tr>
<tr>
<td>Root (Klasse)</td>
<td>56, 57, 97</td>
</tr>
<tr>
<td>RootDecoder (Klasse)</td>
<td>57</td>
</tr>
<tr>
<td>RootIbm2Decoder (Klasse)</td>
<td>58</td>
</tr>
<tr>
<td>RootLM (Klasse)</td>
<td>57, 58</td>
</tr>
<tr>
<td>RootTM (Klasse)</td>
<td>57, 58</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Satz</td>
<td>36, 37</td>
</tr>
<tr>
<td>Satzperplexität †Phrasenperplexität</td>
<td></td>
</tr>
<tr>
<td>saveTo (Methode)</td>
<td>94</td>
</tr>
<tr>
<td>Schaff, Thomas</td>
<td>18</td>
</tr>
<tr>
<td>Schablomenmethode †Template Methode</td>
<td></td>
</tr>
<tr>
<td>SDU †Sematical Dialog Unit</td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td>35, 79</td>
</tr>
<tr>
<td>Sematical Dialog Unit</td>
<td>32</td>
</tr>
<tr>
<td>Sementik</td>
<td>32</td>
</tr>
<tr>
<td>set †Wert, Liste</td>
<td></td>
</tr>
<tr>
<td>setDecoder (Methode)</td>
<td>56, 58</td>
</tr>
<tr>
<td>setErr (Funktion)</td>
<td>93</td>
</tr>
<tr>
<td>setLexicon (Methode)</td>
<td>55, 58</td>
</tr>
<tr>
<td>setLog (Funktion)</td>
<td>93</td>
</tr>
<tr>
<td>setOut (Funktion)</td>
<td>93</td>
</tr>
<tr>
<td>Software</td>
<td>105</td>
</tr>
<tr>
<td>Speaker †Sprecher</td>
<td>32</td>
</tr>
<tr>
<td>Speech Act</td>
<td></td>
</tr>
<tr>
<td>SpeechActPart (Klasse)</td>
<td>57</td>
</tr>
<tr>
<td>Sprachbarriere</td>
<td>15</td>
</tr>
<tr>
<td>Sprache</td>
<td></td>
</tr>
<tr>
<td>Baum</td>
<td>37</td>
</tr>
<tr>
<td>Term</td>
<td>37</td>
</tr>
<tr>
<td>Spracherkennung</td>
<td>21</td>
</tr>
<tr>
<td>Sprachmodell</td>
<td>20–23, 37, 55, 75</td>
</tr>
<tr>
<td>Addition</td>
<td>55</td>
</tr>
<tr>
<td>für Terme</td>
<td>38–40</td>
</tr>
<tr>
<td>Sprachverstecken †Verstehen</td>
<td></td>
</tr>
<tr>
<td>Sprecher</td>
<td>32</td>
</tr>
<tr>
<td>statistische Übersetzung</td>
<td></td>
</tr>
<tr>
<td>†Übersetzung, statistische</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>53, 93</td>
</tr>
<tr>
<td>Stelligkeit</td>
<td>37</td>
</tr>
<tr>
<td>string2int (Methode)</td>
<td>54</td>
</tr>
<tr>
<td>string2phrase (Methode)</td>
<td>54</td>
</tr>
<tr>
<td>String (Datentyp)</td>
<td>53</td>
</tr>
<tr>
<td>StringIterator (Klasse)</td>
<td>55, 97</td>
</tr>
<tr>
<td>StringIteratorAdapter (Klasse)</td>
<td>97</td>
</tr>
<tr>
<td>Struktur, propositionale</td>
<td>31</td>
</tr>
<tr>
<td>strukturelle Divergenzen</td>
<td>50</td>
</tr>
<tr>
<td>Suche †Decoding</td>
<td></td>
</tr>
<tr>
<td>Suchfehler</td>
<td>62</td>
</tr>
<tr>
<td>Sun Microsystems</td>
<td>91</td>
</tr>
<tr>
<td>Symbol</td>
<td>37, 107</td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
<tr>
<td>†(Methode)</td>
<td>55</td>
</tr>
<tr>
<td>Template Method</td>
<td>101</td>
</tr>
<tr>
<td>Term</td>
<td>37, 75, 107</td>
</tr>
<tr>
<td>Blatt</td>
<td>37</td>
</tr>
<tr>
<td>Breite</td>
<td>37</td>
</tr>
<tr>
<td>Ebene</td>
<td>38</td>
</tr>
<tr>
<td>Kopf eines</td>
<td>38</td>
</tr>
<tr>
<td>Stelligkeit</td>
<td>37</td>
</tr>
<tr>
<td>Tiefe</td>
<td>38</td>
</tr>
<tr>
<td>Unter</td>
<td>37</td>
</tr>
<tr>
<td>Variable</td>
<td>37</td>
</tr>
<tr>
<td>Wurzel</td>
<td>37</td>
</tr>
<tr>
<td>Termersetzung</td>
<td>41</td>
</tr>
<tr>
<td>Termersetzungssystem</td>
<td>39</td>
</tr>
<tr>
<td>Termordnung</td>
<td>39</td>
</tr>
<tr>
<td>Termsprache</td>
<td>37</td>
</tr>
<tr>
<td>Tetragramm</td>
<td>59</td>
</tr>
<tr>
<td>Textrepräsentation</td>
<td>31</td>
</tr>
<tr>
<td>Thiele, Frederik</td>
<td>18</td>
</tr>
<tr>
<td>Tiefe</td>
<td>107</td>
</tr>
<tr>
<td>Button-up- †Button-Up-Tiefe eines Terms</td>
<td></td>
</tr>
<tr>
<td>Tiefe</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Seitenzahl</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>time</td>
<td>53</td>
</tr>
<tr>
<td>Top-Down-Ebene</td>
<td>38</td>
</tr>
<tr>
<td>Top-Down-Tiefe</td>
<td>38</td>
</tr>
<tr>
<td>Training</td>
<td>20</td>
</tr>
<tr>
<td>trans</td>
<td>55, 56, 58</td>
</tr>
<tr>
<td>TranslationModel</td>
<td>54, 55, 57, 97</td>
</tr>
<tr>
<td>travel planning</td>
<td>Reiseplanung</td>
</tr>
<tr>
<td>Tree</td>
<td>56, 57, 97</td>
</tr>
<tr>
<td>TreeDecoder</td>
<td>57</td>
</tr>
<tr>
<td>TreeLbm1TM</td>
<td>57, 58</td>
</tr>
<tr>
<td>TreeLbm2Decoder</td>
<td>57, 58</td>
</tr>
<tr>
<td>TreeLbm2TM</td>
<td>57</td>
</tr>
<tr>
<td>TreeLM</td>
<td>57, 58</td>
</tr>
<tr>
<td>TreeSibm2TM</td>
<td>57</td>
</tr>
<tr>
<td>TreeThm2TM</td>
<td>58</td>
</tr>
<tr>
<td>TreeTM</td>
<td>57</td>
</tr>
<tr>
<td>Trigramm</td>
<td>22, 59</td>
</tr>
<tr>
<td>Tupel</td>
<td>101–104</td>
</tr>
<tr>
<td>TupleMap</td>
<td>94–96</td>
</tr>
<tr>
<td>Turn</td>
<td>79</td>
</tr>
<tr>
<td>type</td>
<td>53</td>
</tr>
<tr>
<td>Typumwandlung</td>
<td>53</td>
</tr>
<tr>
<td>Übersetzung</td>
<td>67</td>
</tr>
<tr>
<td>acceptabel</td>
<td>67</td>
</tr>
<tr>
<td>bad</td>
<td>67</td>
</tr>
<tr>
<td>linguistische</td>
<td>15, 31–36</td>
</tr>
<tr>
<td>okay</td>
<td>67</td>
</tr>
<tr>
<td>perfekte</td>
<td>67</td>
</tr>
<tr>
<td>statistische</td>
<td>15, 19–30</td>
</tr>
<tr>
<td>Übersetzungsmodell</td>
<td>20, 24–28, 40–43, 55, 75</td>
</tr>
<tr>
<td>Addition</td>
<td>55</td>
</tr>
<tr>
<td>IBM</td>
<td>IBM-Modelle</td>
</tr>
<tr>
<td>Multiplikation</td>
<td>Projektion</td>
</tr>
<tr>
<td>Projektion</td>
<td>Projektion</td>
</tr>
<tr>
<td>Übersetzungswahrscheinlichkeit</td>
<td>25, 107</td>
</tr>
<tr>
<td>Übertrainieren</td>
<td>71</td>
</tr>
<tr>
<td>unbekanntes Wort</td>
<td>54, 63</td>
</tr>
<tr>
<td>Unigramm</td>
<td>50</td>
</tr>
<tr>
<td>(unk)</td>
<td>Wortklasse, (unk)</td>
</tr>
<tr>
<td>unknown word</td>
<td>unbekanntes Wort</td>
</tr>
<tr>
<td>Unruh, Dominique</td>
<td>18</td>
</tr>
<tr>
<td>Unterterm</td>
<td>37</td>
</tr>
<tr>
<td>updateDescriptor</td>
<td>Methode</td>
</tr>
<tr>
<td>Value</td>
<td>Wert od. Kopf</td>
</tr>
<tr>
<td>Value</td>
<td>Klasse</td>
</tr>
<tr>
<td>Variable</td>
<td>37</td>
</tr>
<tr>
<td>Varianz</td>
<td>50</td>
</tr>
<tr>
<td>Verbform</td>
<td>105</td>
</tr>
<tr>
<td>Verkettung</td>
<td>Projektion</td>
</tr>
<tr>
<td>Vernetzung</td>
<td>15</td>
</tr>
<tr>
<td>verrauchter Kanal</td>
<td>19–20</td>
</tr>
<tr>
<td>Verstehen</td>
<td>15, 16, 31, 77</td>
</tr>
<tr>
<td>Vogel, Stephan</td>
<td>18, 83, 105</td>
</tr>
<tr>
<td>Voraktivierung</td>
<td>21</td>
</tr>
<tr>
<td>Wahrscheinlichkeit</td>
<td>107</td>
</tr>
<tr>
<td>bedingte</td>
<td>107</td>
</tr>
<tr>
<td>Waibel, Alex</td>
<td>18</td>
</tr>
<tr>
<td>Wallace, Dorcas</td>
<td>18</td>
</tr>
<tr>
<td>Wang, Ye-Yi (王野翊)</td>
<td>29, 30, 55</td>
</tr>
<tr>
<td>Weaver, Warren</td>
<td>15, 19</td>
</tr>
<tr>
<td>Welt</td>
<td>77</td>
</tr>
<tr>
<td>Wert</td>
<td>32, 99</td>
</tr>
<tr>
<td>einfachter</td>
<td>33</td>
</tr>
<tr>
<td>Klasse</td>
<td>99</td>
</tr>
<tr>
<td>komplexer</td>
<td>33</td>
</tr>
<tr>
<td>Liste</td>
<td>34</td>
</tr>
<tr>
<td>Menge</td>
<td>Wert Liste</td>
</tr>
<tr>
<td>Werteklasse</td>
<td>99</td>
</tr>
<tr>
<td>Wiedenhoff, Christian</td>
<td>18</td>
</tr>
<tr>
<td>Word Error Rate</td>
<td>Wortfehlerate</td>
</tr>
<tr>
<td>Wort, leerer</td>
<td>24</td>
</tr>
<tr>
<td>Wortfehlerate</td>
<td>72, 107</td>
</tr>
<tr>
<td>Wortfolge</td>
<td>107</td>
</tr>
<tr>
<td>Wortform</td>
<td>37, 54</td>
</tr>
<tr>
<td>Wortklasse</td>
<td>28, 76–77, 107</td>
</tr>
<tr>
<td>(first-name)</td>
<td>76</td>
</tr>
<tr>
<td>(last-name)</td>
<td>76</td>
</tr>
<tr>
<td>(unk)</td>
<td>54, 64</td>
</tr>
<tr>
<td>Writeable (Klasse)</td>
<td>93, 95</td>
</tr>
<tr>
<td>Wurzel</td>
<td>37, 47</td>
</tr>
<tr>
<td>WWDecoder (Klasse)</td>
<td>54–56</td>
</tr>
<tr>
<td>Zahl</td>
<td>99</td>
</tr>
<tr>
<td>Zielsprache</td>
<td>19</td>
</tr>
<tr>
<td>Zugriffspfad</td>
<td>41</td>
</tr>
<tr>
<td>Zuordnung</td>
<td>16, 17, 24, 55, 59–62, 76</td>
</tr>
</tbody>
</table>
Zuordnungswahrscheinlichkeit ... 26, 41, 50, 107
Hilfsverteilung ...................... 51
Zusammenschaltung  ↑ Projektion

Zuweisung  .................................. 92
Zwischensprache  .......................... 32