Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Logik, Komplexität und Deduktionssysteme

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.

Anders als herkömmliche Verfahren, die hauptsächlich auf regelbasierten Analysetechniken aufbauen, ist das Verfahren, das in dieser Arbeit entwickelt wird, eine Adaption von Verfahren, die man in der statistischen Maschinenübersetzung einsetzt. Statt daß man derartige Verfahren auf ein Paar von natürlichen Sprachen anwendet, um damit z. B. von Deutsch nach Englisch zu übersetzen, betrachten wir die zwischensprachliche Repräsentation als Zielsprache eines statistischen Übersetzers und beschreiben Sprachverstehen als „Übersetzung" von einer natürlichen Quellsprache in diese Zielsprache. Da die meisten gängigen Zwischensprachen, insbesondere das am Lehrstuhl verwendete *Interchange Format*, baumartig strukturiert sind, während die klassischen Übersetzungsmodelle Sätze als lineare Ketten von Wörtern auffassen, widmet sich der zentrale Teil der Arbeit der nötigen Verallgemeinerung vorhandener statistischer Modelle und Algorithmen auf derartig strukturierte Gebilde.

Ein wesentlicher Vorteil, den automatische trainierbare Verfahren gegenüber handgeschriebenen Regelsystemen haben, ist ökonomischer Natur. So kann ein allgemein implementiertes statistisches System innerhalb weniger Minuten Parameter trainieren, mit deren Hilfe Ergebnisse erzielt werden können, deren Qualität vergleichbar ist mit der Qualität, die durch Grammatiken erzielt wurde, die unter großem Aufwand von Hand geschrieben wurden. In dieser Rechnung wird allerdings unterschlagen, daß zum Training der Parameter ein möglichst großes paralleles Corpus zur Verfügung stehen muß. Da darin für unseren Fall den natürlichsprachlichen Äußerungen Entsprechungen im Interchange Format gegenübergestellt werden müssen, sind für die Zusammenstellung des Corpus weiterhin Experten nötig. Dies motiviert weiterführende Überlegungen, die darauf abzielen, die nötigen Parameter aus einem zweisprachigen Corpus zu extrahieren. Die Idee ist dabei, ein Modell beispielsweise für Deutsch nach Interchange Format aus zwei vorhandenen für Deutsch nach Englisch und Englisch nach Interchange Format zu erhalten.

Die gewonnenen Erkenntnisse wurden zu Evaluationszwecken in einem Softwaresystem implementiert, so daß ein Vergleich zu etablierten Verfahren möglich wurde. 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 Fragestellungen aufgeworfen, die interessante Möglichkeiten zu weiteren Forschungen aufzeigen.

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 terminologues 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 natura lingvo. En la kunteksto de komputilsistemoj tio signifas kreadon de lingvosendependa reprezento de la semantiko de naturlingva teksto.

La metodo, kiu elvolvatas en ĉi-laboro estas – alimaniere ol la kutime uzataj, kiuj ĉefe uzas regulbazitajn analizteknikojn – adapto 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 konsideras la interlingvan reprezenton kvazaŭ cellingvo de la statistika traduksistemo kaj la lingvokomprenon kvazaŭ traduko de naturlingva fontlingvo 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 necesa ĝeneraligo de ekzistantaj statistikaj modeloj kaj algoritmoj al tiaj strukturoj.

Ĉefa avantaĝo de aŭtomate trejneblaj metodoj kompare kun mane kreitaj regulsistemoj estas ekonoma: Statistika sistemo povas – unufoje programita – dum kelkaj minutoj lerni parametrojn, kun kiuj eblas ricevi rezultojn, kiuj estas rilate al la kvalito kompareblaj kun tiuj de sistemoj uzantaj gramatikojn – mankreitajn 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 konsistas el paroj de naturlingvaj frazoj kaj la respektivaj tradukoj al la Interchange Format, oni ankaŭ ĉi tie bezonas ekspertojn por la kolekto de la korpuso. Tio motivigas pluajn konsiderojn, celante al lernado de la necesaj parametroj el korpuso kun du naturaj lingvoj. La ideo estas, ke modelo povus krei korpuson por la Germana kaj Interchange Format el du korpusoj: German-Angla kaj korpuso por la Angla kaj Interchange Format.

La ekkonoj trovitaj dum nia laboro realigatis en programo por ebligi komparon kun kutime uzataj metodoj. Tiel evidentiĝis, ke – malgraŭ la kompare simpla metodo kaj la nur malgranda trejnkorpuso – riceviĝis surprize bonaj rezultoj. Samtempe la rezultoj de la eksperimentoj aperigis novajn demandojn, kiuj indikas interesajn eblecojn por plua esplorado.

Inhaltsverzeichnis

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig und ohne unerlaubte Hilfe verfaßt habe. Alle wörtlichen und sinngemäßen Zitate sind als solche kenntlich gemacht.

Karlsruhe, 31. Mai 2002

Manuel Kauers

Manuel Kauers

1 Einleitung

> Die Mathematiker sind eine Art Franzosen, redet man zu ihnen, so übersetzen sie es in ihre Sprache und dann ist es alsobald ganz etwas anderes.
>
> JOHANN WOLFGANG VON GOETHE

Das Problem der automatisierten Sprachübersetzung ist für die Informatik aus unterschiedlichen Gründen interessant. Auf der einen Seite wächst durch die fortschreitende weltweite Vernetzung der Völker der Wunsch nach einer rechnergestützten Überbrückung von Sprachbarrieren. Obwohl heutige Systeme einen verschwindend geringen Marktanteil in der Übersetzung haben, ist anzunehmen, daß der potentielle Markt für „gute" Systeme weitaus größer ist. Auf der anderen Seite ist Sprachübersetzung auch rein informatisch betrachtet interessant. Neben technischen Fragestellungen spielen dabei das Problem der mathematischen Faßbarkeit (bzw. Unfaßbarkeit) von Sprache sowie Überlegungen, die weit in Linguistik und Philosophie reichen, eine Rolle.

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 $(\mathbf{e}, \mathbf{f})$ von Sätzen eine approximierte Wahrscheinlichkeit dafür angegeben wird, daß $\mathbf{e}$ eine gültige Übersetzung des Satzes $\mathbf{f}$ ist. Der Vorgang des Übersetzens besteht dann in der Suche nach einem Satz $\mathbf{e}$, für den die Wahrscheinlichkeit $p(\mathbf{e}|\mathbf{f})$ zu vorgegebenem $\mathbf{f}$ 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 Übersetzungsvorgang 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 natürlichsprachlichen 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. Einzige Bedingung ist, daß ein genügend großes bilinguales Corpus für das gewünschte Sprachpaar zur Verfügung steht.

Flexibilität, wenngleich in einem anderen Sinne, wird auch als typische Eigenschaft linguistischer Methoden ins Feld geführt. Sie wird vor allem wichtig, wenn mit einem System mehr als zwei Sprachen verarbeitet werden sollen. Durch Verwendung einer standardisierten Zwischenrepräsentation (*Interlingua*) braucht für eine zusätzliche Sprache nur je ein Modul für die Überführung natürlicher Sprache in diese Zwischenrepräsentation und für die Generierung natürlichsprachlicher Formulierungen aus der Zwischenrepräsentation zur Verfügung gestellt zu werden. Übersetzer für ein konkretes Sprachpaar erhält man dann durch Hintereinanderschaltung des „Verstehers" der Quellsprache und des „Generierers" der Zielsprache. Statt daß man für jedes der $O(n^2)$ Sprachpaare ein eigenes Modul bereitstellen muß, genügen dabei nur $O(n)$ Module (Abbildung 1).

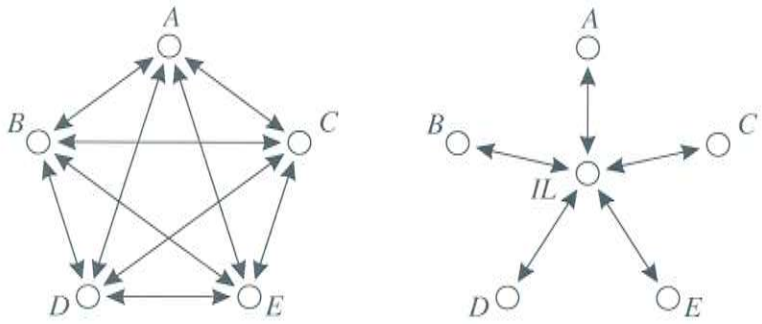

Abbildung 1 Veranschaulichung der Komplexitätsreduktion beim Einsatz einer Interlingua

Nachdem man zum Sprachverstehen bisher verstärkt regelbasierte Systeme eingesetzt hatte, wurden in jüngster Zeit Versuche unternommen, auch die Verfahren der statistischen Übersetzung zum Sprachverstehen einzusetzen. Macherey *et al.* (2001) wendeten dazu ein bestehendes statistisches Übersetzungssystem nicht wie sonst üblich auf ein Paar natürlicher Sprachen an, sondern entwickeln eine semantische Repräsentation und trainieren die Übersetzung von Deutsch in diese Darstellung. Die dabei verwendete Kunstsprache basiert auf Schlüsselwörtern für *Konzepte*, die durch die Quellsatzwörter, denen sie zugeordet werden, näher spezifiziert werden. Abbildung 2 zeigt ein Beispiel. Durch explizite Berechnung der Zuordnung erhält man dort *Köln* als „Argument" von `@origin` und *München* als Argument von `@destination`.

Die lineare Form der Konzeptketten ermöglichte 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.[1] 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

[1] Zwar 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.

Motiviert werden Untersuchungen in diese Richtung durch die Probleme, die man bei der Erstellung von Regelsystemen bekommt. Diese Probleme sind sowohl technischer wie auch ökonomischer Natur. Die Hoffnung ist, daß sich der Einsatz statistischer Verfahren hier als vorteilhaft erweist, da es dabei nur nötig ist, Parameter zu allgemein formulierten Modellen anhand eines Trainingscorpus zu schätzen. Dabei ist zu beachten, daß die Trainingsmenge in unserem Fall nicht äquivalente Aussagen in zwei natürlichen Sprachen gegenüberstellt, sondern daß vielmehr natürlichsprachlichen Äußerungen entsprechende Interlingua-Repräsentation gegenüberzustellen sind.

Zwar ist zu vermuten, daß die Annotation einer Trainingmenge 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.

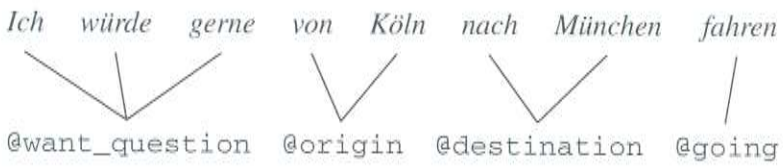

Abbildung 2 Eine Zuordnung von Wörtern eines deutschen Quellsatzes zu Konzeptbezeichnern, vgl. Macherey *et al.* (2001), Abbildung 1

Übersicht

Die vorliegende Diplomarbeit ist wie folgt aufgebaut:

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

Kapitel 4 entwickelt die nötigen Verallgemeinerungen der statistsichen Modelle aus Kapitel 2 zur Übersetzung natürlicher Sprache in Termsprachen. Diese Modelle werden zunächst für allgemeine Termsprachen formuliert und später auf das im Nespole!-Projekt verwendete *Interchange Format* angewandt.

Kapitel 5 beschäftigt sich im Anschluß daran mit der Projektion eines natürlichsprachlichen Übersetzungsmodells 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 Schaaf und Christian Wiedenhoff, die als Humanevaluatoren zur Verfügung standen.

Schließlich danke ich Christian Fügen, Jürgen Reichert, Thomas Schaaf und besonders Stephan Vogel und Alex Waibel für ihre inhaltlichen Beiträge und ihre Diskussionsfreudigkeit.

2 Statistische Methoden

2.1 Das Modell des verrauschten Kanals

Als Warren Weaver vorschlug, Methoden aus der statistischen Kryptoanalyse auch zur automatischen Übersetzung natürlicher Sprache zu verwenden (Hutchings, 1997), erlaubten die technischen Gegebenheiten es noch nicht, Corpora hinreichender Größe zu verarbeiten. Doch im Zuge sich ständig vergrößernder Speicher- und Rechenkapazitäten heutiger Computeranlagen rücken statistische Ansätze bei der Übersetzung natürlicher Sprache wieder verstärkt ins Interesse. Grundlage der meisten modernen statistischen Modelle ist die Arbeit von Brown *et al.* (1993), die Weavers konzeptionelle Idee, Sprachübersetzung mit Hilfe eines *verrauschten Kanals* zu modellieren, in mathematische Formeln kleidet und Trainingsalgorithmen angibt. Diesem Artikel folgt die Darstellung des vorliegenden Kapitels.

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 „begreifen", um später aus dem gestörten Signal (d. h. aus dem Quellsatz) das ungestörte Ursprungssignal (d. h. den Zielsatz) zu rekonstruieren.

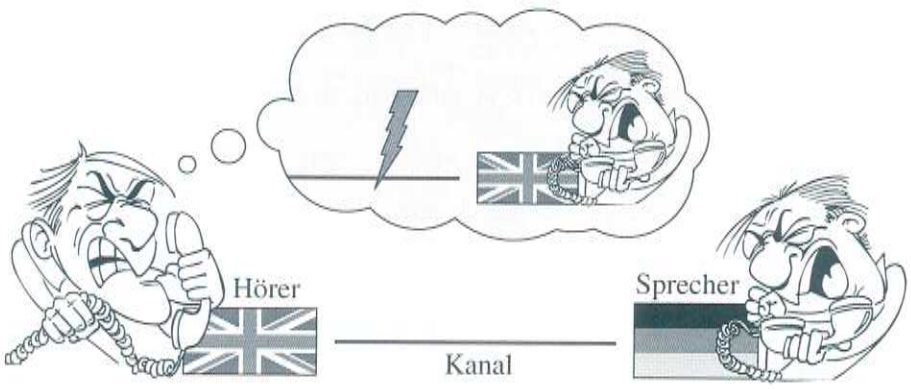

Abbildung 3 Das Modell des verrauschten Kanals: Der Hörer, dessen Sprache (hier Englisch) von der des Sprechers (hier Deutsch) verschieden ist, glaubt, der Sprecher spräche sehr wohl die gleiche Sprache wie er. Daß er den Sprecher dennoch nicht versteht, erklärt er sich durch Störungen des Übetragungskanals.

Bezeichnet man mit $p(\mathbf{e}|\mathbf{f})$ die Wahrscheinlichkeit dafür, daß ein menschlicher Experte den Satz $\mathbf{e}$ als Übersetzung des Satzes $\mathbf{f}$ nennt, dann läßt sich Übersetzung beschreiben als die *Suche* nach einem Satz $\hat{\mathbf{e}}$, so daß für vorgegebenes $\mathbf{f}$ die Wahrscheinlichkeit $p(\hat{\mathbf{e}}|\mathbf{f})$ maximal wird:

$$\hat{\mathbf{e}} := \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}).$$

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

$$p(\mathbf{e}|\mathbf{f}) = \frac{p(\mathbf{e})}{p(\mathbf{f})} p(\mathbf{f}|\mathbf{e}),$$

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

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \left(p(\mathbf{e}) p(\mathbf{f}|\mathbf{e}) \right). \tag{1}$$

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

$$p(\textit{good morning , sir .}) > p(\textit{morning , sir . good}).$$

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

$$\begin{aligned} &p(\textit{Guten Morgen .} \mid \textit{good morning , sir .}) \\ &> p(\textit{Auf Wiedersehen .} \mid \textit{good morning , sir .}) \end{aligned}$$

und spricht bei einer Modellierung dieser Wahrscheinlichkeit vom *Übersetzungsmodell*.[3]

Es bleibt also nunmehr das Problem, allgemeine Sprach- und Übersetzungsmodelle zu formuleren, deren Parameter sich durch Analyse eines bilingualen Corpus algorithmisch schätzen lassen (*Training*). Damit beschäftigen sich die folgenden beiden Abschnitte. Ein weiteres Problem beschert der Suchraum: Zu jedem Quellsatz $\mathbf{f}$ gibt es unendlich viele – bei Beschränkung der Satzlänge immer noch exponentiell viele – potentielle Übersetzungen $\mathbf{e}$, unter denen die optimale Übersetzung $\hat{\mathbf{e}}$ 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 Interpunktionszeichen wie einzelne Wörter behandelt.

2.2 Ein einfaches Sprachmodell

Sprachmodelle werden nicht nur bei der statistischen Übersetzung, sondern auch in der Spracherkennung verwendet. Die Verfeinerung und Verbesserung althergebrachter Sprachmodell-Ansätze ist daher Thema zahlreicher wissenschaftlicher Arbeiten gewesen. Einen Überblick über Standardtechniken findet man bei Jelinek (1999). Für unsere Zwecke soll jedoch ein vergleichsweise einfaches Sprachmodell genügen, da mühsam gewonnene Verbesserungen häufig doch niedriger sind als das Rauschen der Übersetzungswahrscheinlichkeit und folglich untergehen würden. Insbesondere soll auf Verfeinerungen verzichtet werden, die spezielles Wissen über die modellierte Sprache ausnutzen, da das im folgenden dargestellte Sprachmodell später (Kapitel 4) auf Termsprachen übertragen werden soll, deren Struktur sich erheblich von der Struktur natürlicher Sprachen unterscheidet.

Die Grundlage der meisten Sprachmodelle ist die Zerlegung

$$p(\mathbf{e}) = \prod_{i=1}^{l} p(e_i|e_1,\ldots,e_{i-1}), \tag{2}$$

nach der die Wahrscheinlichkeit für eine Sequenz $\mathbf{e} = (e_1,\ldots,e_l)$ 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 Sprachverarbeiter eine Erwartungshaltung darüber 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 Specher 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 macht, 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

$$p(e_i|e_1,\ldots,e_{i-1}) :\approx p(e_i|e_{i-n},\ldots,e_{i-1}) := \frac{n(e_{i-n},\ldots,e_i)}{\sum_e n(e_{i-n},\ldots,e_{i-1},e)}$$

aller im Trainingsmaterial gesehenen n-Gramme $(e_{i-n},\ldots,e_i)$. Steht hinreichend viel Trainingsmaterial zur Verfügung, so steigt mit wachsendem n (neben 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. Nähme 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. ungesehene Trigramme keine Seltenheit sind. Grob vereinfachend nehme man dazu an, daß alle 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 α steigt, sind folglich

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

Ziehungen nötig. Legt man nun eine realistische Lexikongröße von 10^5 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.[4] Konkrete Untersuchungen am Nespole!-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 α der Wahrscheinlichkeitsmasse für ungesehene n-Gramme zurückgehalten. Man erhält dann

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

Formal betrachtet man alle ungesehenen (bzw. allgemeiner: selten gesehenen) Tupel als gleich und ordnet diesen die zurückgehaltene Wahrscheinlichkeit α 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:

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

[4] 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 $\sum_i \lambda_i = 1$.

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

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

Durch sie wird angegeben, wie groß eine als gleichverteilt angenommene Referenzmenge sein müßte, damit dort die zufällige Wahl eines Elements die gleiche Unsicherheit hat wie eine Vorhersage des Sprachmodells in der modellierten Sprache. Bei einer minimalen Perplexität von 1 ist demnach jede Vorhersage des Sprachmodells ein Treffer, während mit zunehmender Perplexität die Unsicherheit des Sprachmodells wächst. Vereinfacht ausgedrückt läßt sich die Perplexität auffassen als die mittlere Anzahl der voraktivierten Wörter.

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 ungesehenen Menge B minimiert wird. Dazu bietet sich eine rekursive Formulierung der Interpolation an:

$$\begin{aligned} p''(e_i) &:= p'(e_i), \\ p''(e_i|e_{i-n}, \ldots, e_{i-1}) &:= \lambda_n p'(e_i|e_{i-n}, \ldots, e_{i-1}) + (1-\lambda_n) p'(e_i|e_{i-n+1}, \ldots, e_{i-1}), \end{aligned}$$

über die sich λ_1 bis λ_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

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

ergeben. Für jede dieser Zerlegungen werden unabhängig voneinander Interpolationsparameter $\lambda_1^{(i)}, \ldots, \lambda_n^{(i)}$ ermittelt, deren arithmetisches Mittel zu den schließlich verwendeten Werten führt:

$$\lambda_k := \frac{1}{m} \sum_{i=1}^{m} \lambda_k^{(i)} \qquad (k = 1, \ldots, n).$$

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

Für Einzelheiten und weitere Verbesserungen sei nochmals auf Jelinek (1999) verwiesen.

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 $(\mathbf{e},\mathbf{f})$, 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.

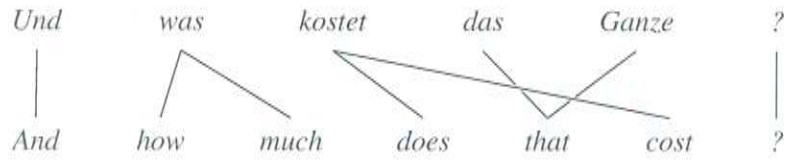

Abbildung 4 Beispielhafte Zuordnung zwischen den Wörtern zweier entsprechender Sätze des Deutschen und Englischen

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 $\mathbf{f}$ der Länge m und einem anderen Satz $\mathbf{e}$ der Länge l als Funktion $\mathbf{a}\colon \{1,\ldots,m\} \to \{0,\ldots,l\}$ und sagt, die Positionen i und j seien einander zugeordnet, falls $a_j := \mathbf{a}(j) = i$ gilt. Im Fall $a_j = 0$ sei der j-ten Position kein Wort der anderen Seite zugeordnet. (Jeder Satz $\mathbf{e} = (e_1,\ldots,e_l)$ wird dazu formal um ein *leeres Wort* an Position $i = 0$ ergänzt.)

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

$$p(\mathbf{f}|\mathbf{e}) := \sum_{\mathbf{a}} p(\mathbf{f},\mathbf{a}|\mathbf{e})$$

mit

$$p(\mathbf{f},\mathbf{a}|\mathbf{e}) = p(m|\mathbf{e}) \prod_{j=1}^{m} \left(p(a_j|a_1^{j-1}, f_1^{j-1}, m, \mathbf{e}) p(f_j|a_1^{j}, f_1^{j-1}, m, \mathbf{e}) \right). \tag{3}$$

Dabei steht x_y^z kurz für $x_y,\ldots,x_z$. Diese Zerlegung fußt auf der Vorstellung, daß man, $\mathbf{f}$ aus $\mathbf{e}$ generierend, zunächst die Länge m von $\mathbf{f}$ bestimmt, um dann für jedes $j = 1,\ldots,m$ zunächst die zugeordnete Zielposition 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 $\mathbf{f}$ aus $\mathbf{e}$ generiert wird, während wir insgesamt an der umgekehrten Richung interessiert sind.

Übersetzungswahrscheinlichkeiten (engl. *translation probabilities*) $t(f|e)$:

$$p(\mathbf{f}|\mathbf{e}) := \sum_{\mathbf{a}} \frac{1}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{a_j}) \tag{4.a}$$

$$= \frac{1}{(l+1)^m} \sum_{a_1=0}^{l} \sum_{a_2=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(f_j|e_{a_j}) \tag{4.b}$$

$$\overset{\text{(Brown } et\ al.\text{, 1993)}}{=} \frac{1}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i) \tag{4.c}$$

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 $p(\mathbf{f}|\mathbf{e})$ für vorgegebene $(\mathbf{f},\mathbf{e})$ 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

$$h(t(f|e),\lambda) = p(\mathbf{f}|\mathbf{e}) - \sum_{e'} \lambda_{e'} \left(\sum_{f'} t(f'|e') - 1 \right)$$

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 $g_i(x) \neq 0$) und verwendet die Darstellung (4.c) von $p(\mathbf{f}|\mathbf{e})$, so erhält man

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

$$\iff \qquad t(f|e) = \lambda_e^{-1} p(\mathbf{f}|\mathbf{e}) \frac{t(f|e)}{\sum_{i=0}^{l} t(f|e_i)} \sum_{j=1}^{m} \delta(f,f_j) \sum_{i=0}^{l} \delta(e,e_i). \tag{5}$$

Die durch

$$c(f|e,\mathbf{f},\mathbf{e}) := \frac{t(f|e)}{\sum_{i=0}^{l} t(f|e_i)} \sum_{j=1}^{m} \delta(f,f_j) \sum_{i=0}^{l} \delta(e,e_i) \overset{(5)}{=} \lambda_e \frac{t(f|e)}{p(\mathbf{f}|\mathbf{e})}$$

definierten *Counts* lassen sich interpretieren als Erwartungswert für die Anzahl der Verbindungen zwischen Wörtern f und e im Satzpaar $(\mathbf{f},\mathbf{e})$. 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

$$t(f|e) = \lambda'_e \sum_{(\mathbf{f},\mathbf{e})\in C} c(f|e,\mathbf{f},\mathbf{e})$$

für geeignete Normierungskonstanten λ'_e.

Da die Counts $c(f|e,\mathbf{f},\mathbf{e})$, 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     forall e, f do    // initialize t arbitrarily
5       t(f|e) := 1/L.size();
6     for n = 1 to number of iterations do
7       forall e, f do    // initialize counts to zero
8         c(e, f) := 0;
9       forall (e,f) ∈ Corpus do    // determine counts
10        for j = 1 to m do
11          sum := Σ_{i=0}^{l} t(f_j|e_i);
12          for i = 0 to l do
13            c(e_i, f_j) := c(e_i, f_j) + t(f_j|e_i)/sum;
14      forall e, f ∈ L do    // redefine translation probs as normalized counts.
15        t(f|e) := λ'_e c(e, f);
16    return t;
17  end
```

Algorithmus 1 Algorithus zum Training der Parameter des IBM 1 Übersetzungsmodells

Das Modell IBM 2 verwendet neben den Übersetzungswahrscheinlichkeiten positionsabhängige *Zuordnungswahrscheinlichkeiten* (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 $\mathbf{f}$ dem i-ten Wort in $\mathbf{e}$ zuzuordnen ist. Damit ergibt sich

$$p(\mathbf{f}|\mathbf{e}) := \sum_{\mathbf{a}} \prod_{j=1}^{m} t(f_j|e_{a_j}) a(a_j|j,m,l) \tag{6.a}$$

$$= \sum_{a_1=0}^{l} \sum_{a_2=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(f_j|e_{a_j}) a(a_j|j,m,l) \tag{6.b}$$

$$\overset{\text{(Brown et al., 1993)}}{=} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i) a(i|j,m,l) \tag{6.c}$$

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

$$h(t,a,\lambda,\mu) = p(\mathbf{f}|\mathbf{e}) - \sum_{e'} \lambda_{e'} \Big(\sum_{f'} t(f'|e') - 1 \Big) - \sum_{j'} \mu_{jml} \Big(\sum_{i'} a(i'|j',m,l) - 1 \Big).$$

Dies führt mit (6.c) auf

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

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

bzw. für die Zuordnungswahrscheinlichkeiten auf

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

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

Daraus ergeben sich wie oben die Counts

$$c(f|e,\mathbf{f},\mathbf{e}) = \sum_{j=1}^{m} \sum_{i=0}^{l} \frac{t(f|e)a(i|j,m,l)\delta(f,f_j)\delta(e,e_i)}{\sum_{k=0}^{l} t(f|e_k)a(k|j,m,l)} \quad \text{und}$$

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

und damit die Trainingsgleichungen

$$t(f|e) = \lambda'_e \sum_{(\mathbf{f},\mathbf{e})\in C} c(f|e,\mathbf{f},\mathbf{e}), \qquad a(i|j,m,l) = \mu'_{jml} \sum_{(\mathbf{f},\mathbf{e})\in C} c(i|j,m,l,\mathbf{f},\mathbf{e})$$

für geeignet gewählte Normierungskonstanten λ'_e und μ'_{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 gestattet, 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 $n(\varphi_i|e_i)$ dafür eingeführt, daß diesem Wort φ_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.

Die Modelle 3 bis 5 sind hier nur der Vollständigkeit halber erwähnt. Sie spielen in der Arbeit keine weitere Rolle. Der interessierte Leser sei auf Brown *et al.* (1993) verwiesen. Für aktuellere Modelle, die teils mehr, teils weniger stark auf den IBM-Modellen aufbauen, sei zum Beispiel auf Ney *et al.* (2000); Och *et al.* (1999); Vogel *et al.* (1996, 2000); Vogel and Ney (2000); Bangalore and Riccardi (2001) verwiesen.

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 Übersetzungswahrscheinlichkeiten 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

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

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

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

2.4 Ein Algorithmus zum Decoding

Einen Suchalgorithmus für das IBM-2-Modell stellt Wang (1998) vor (vgl. auch Wang and Waibel, 1997). Das Verfahren soll in diesem Abschnitt skizziert werden. Auf sei-

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

Grundidee 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 α übersteigt bzw. die n am besten bewerteten Hypothesen (α und n geeignet gewählt).[6]

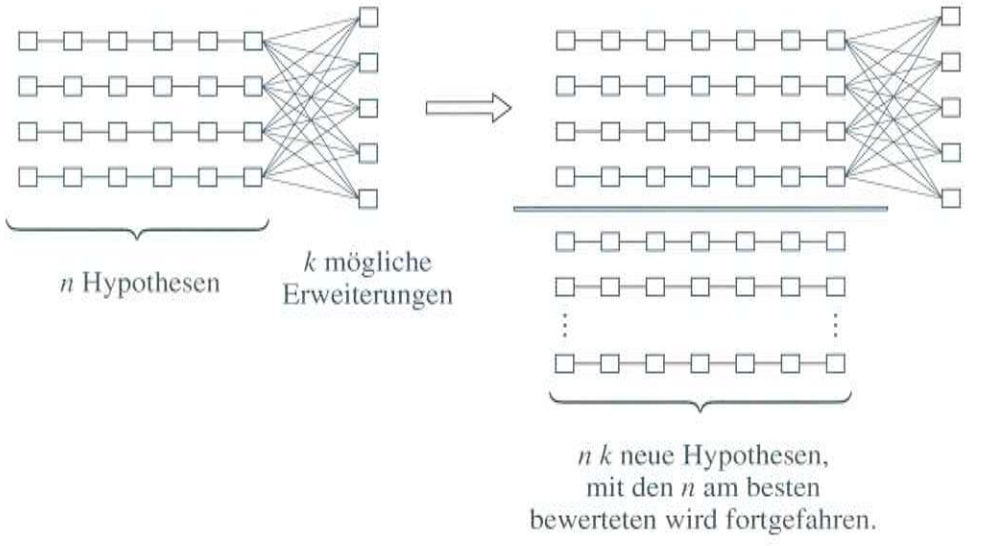

Abbildung 5 Schematische Darstellung der Arbeitsweise des Wang-Decoders

Die Bewertung der Hypothesen orientiert sich an Gleichung (6.c), in der die Wahrscheinlichkeit angegeben ist, die das IBM-2-Modell für $p(\mathbf{f}|\mathbf{e})$ angibt. Während dort die eigentliche Vorstellung ist, daß eine bestimmte Zuordnung $\mathbf{a}$ vorliegt, die die Wörter aus $\mathbf{e}$ zu denen aus $\mathbf{f}$ in Beziehung setzt, interpretiert Wang diese Gleichung so, daß jedes Zielwort e_i zur Generation jedes Quellworts f_j einen gewissen „Anteil", nämlich $t(f_j|e_i)a(i|j,m,l)$, beiträgt. Zur Bewertung einer Hypothese ersetzt er $t(f_j|e_i)$ für diejenigen Wörter e_i, die noch nicht bekannt sind, durch das gewichtete Mittel

$$\sum_e p(e)t(f_j|e).$$

Mit

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

[6] Die erste Variante hat den Vorteil, daß sich die Zahl der behaltenen Hypothesen nach deren Bewertung richtet: Sind viele Hypothesen gut, dann werden auch viele weitergeführt. Der Vorteil der zweiten Variante ist eine bessere Kontrolle des betrachteten Teils des Suchraums. Unsere Implementierung (vgl. Kapitel 6) verwendet eine Kombination der beiden Varianten.

erhält man also

$$\tau(H) = p(e_1,\ldots,e_k) \cdot \sum_{l=k}^{\infty} p(l|m) \prod_{j=1}^{m} \sum_{i=0}^{l} a(i|j,m,l)\,\tau(i,j,H) \tag{7}$$

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:

```
1   function wangDecoding(f)
2     H := {λ};   // set of hypothesises to be extended further
3     R := ∅;   // set of complete results found so far
4     for k = 1 to maximum target sentence length do
5       // generate new hypothesises
6       H_new := ∅;
7       forall h ∈ H do   // for all hypothesises
8         forall e ∈ L_F do   // for all target words
9           H_new := H_new ∪ h.add(e);
10      // cut off less probable hypothesises
11      H := {h ∈ H_new : τ(H) > α};
12      // generate complete results
13      forall h ∈ H do
14        R := R ∪ h.add(end of sentence);
15    // return the most probable translation
16    return arg max_{e∈R} p(e|f);
```

Algorithmus 2 Decoding nach Ye-Yi Wang. Dabei sei L_F das Lexikon der Zielsprache und λ 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 Normalfall 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 geschieht, wenn der Mensch einen Text versteht?

Die Linguistik hat verschiedene Modellvorstellungen entwickelt, die den Prozeß des Sprachverstehens beschreiben. Zu ihnen gehört die Vorstellung von einem *mentalen Modell*, wie es u. a. bei Linke *et al.* (1996) beschrieben wird (dort erläutert am Lesen eines Textes). Demnach handelt es sich beim Sprachverstehen um einen zweistufigen Prozeß.

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 *Präsuppositionen* (implizit vorausgesetzte Bedeutungskomponenten), Deiktika/Pronomen und dergleichen bereits weitgehend aufgelöst, so daß die propositionale Struktur als reichaltiger 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 der wesentliche Inhalt des Textes vollständig repräsentiert ist.

Der Aufbau von Textrepräsentation und mentalem Modell ist als eng zusammenhängend zu verstehen. So können bestehende 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 erneute Interpretationsversuche zu bewahren, wenn sich ihre Aussagen nicht gleich mit dem mentalen Modell in Einklang bringen lassen. Ähnlich werden Textpassagen zum Teil im genauen 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-

stehen im allgemeinen siehe Linke *et al.*, 1996, .)

In seiner Allgemeinheit ist der Begriff des mentalen Modells zwar einerseits sehr intuitiv, andererseits ist jedoch unklar, wie sich eine derart allgemeine Vorstellung mit Hilfe der üblichen Algorithmen und Datenstrukturen im Rahmen eines automatisch arbeitendes Systems implementieren ließe. Da in realistischen Anwendungssituationen die volle Mächtigkeit eines mentalen Modells auch gar nicht nötig ist, beschränkt man sich bei der rechnerinternen Repräsentation von Semantik auf Darstellungen, die sich an den Aufgaben der propositionalen Struktur orientieren. Dies geschieht vornehmlich durch den Einsatz spezieller formaler Kunstsprachen, die im Idealfall eine mehr oder weniger sprachunabhängige semantische Darstellung erlauben. Derartige formale Sprachen bezeichnet man als *Interlingua* oder *Zwischensprache*. Ein Beispiel hierfür sind die auf Seite 16 bereits angesprochenen „Konzeptketten" von Macherey *et al.* (2001). Weitere zum Teil sehr unterschiedliche Ansätze werden u. a. in Lee *et al.* (2001); Levin *et al.* (1998); Nasr *et al.* (1997) vorgestellt.

Die wohl prominenteste Klasse von Zwischensprachen bilden baumartig strukturierte Sprachen. Ein Beispiel für eine solche Sprache ist das *Interchange Format*, das seit geraumer Zeit am Language Technologies Institute der Carnegie Mellon University in Pittsburgh, USA, in Kooperation mit anderen Gruppen entwickelt wird. Dieser Interlingua und ihrer Verwendung zur Sprachübersetzung im Rahmen des Nespole!-Projekts widmen sich die folgenden beiden Abschnitte. Sie wird in den Kapiteln 4 und 7 als exemplarische Interlingua verwendet werden.

3.2 Das Interchange Format

Das *Interchange Format* (IF), die im Nespole!-Projekt (Abschnitt 3.3) verwendete Interlingua, wird u. a. von Levin *et al.* (1998, 2000a,b); Han (2002) beschrieben und untersucht. Dieser Abschnitt soll kurz in das Interchange Format einführen, weiterführende Details entnehme man den oben genannten Arbeiten oder der jeweils aktuellsten Version der Spezifikation (IF, 2002). Den Ausführungen in diesem Abschnitt liegt die Version vom 15. April 2002 zugrunde. (Version der Trainingsmenge A, vgl. 79).

Das Interchange Format beschränkt sich auf den Diskursbereich Reiseplanung und Hotelreservierung. Gespräche in diesem Diskursbereich lassen sich relativ eindeutig in sogenannte *domain actions* (auch *semantical dialog unit*, SDU) zerlegen. Typische solche Aktionen sind grüßen, informieren, fragen, danken oder bitten. Zur vollständigen Beschreibung der Semantik einer SDU verwendet das Interchange Format termartige Gebilde, die sich aus speziellen Symbolen für Sprecher (engl. *Speaker*), Aktionen (engl. *Speech Act*), Konzepten (engl. *Concept*), Argumenten (engl. *Argument*) und Werten (engl. *Value*) zusammensetzen:

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

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 viele eher selten verwendet. Im Nespole!-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 wesentlichen Teile des semantischen Gehalts (etwa Aktionen, Objekte, Attribute, Ereignisse usw.) abgedeckt werden, von denen die Rede ist. Die ca. 110 möglichen Konzeptbezeichner sind nicht beliebig kombinierbar, sie müssen zum Speech Act bzw. zum vorangehenden Konzept passen. Per Spezifikation *lizensieren* Speech Acts und Konzepte, welche Konzepte ihnen folgen dürfen. Zur formalen Unterscheidung von Konzept- und Aktionsbezeichnern beginnen Konzeptbezeichner mit dem Symbol „+". Im Nespole!-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. Argumentbezeichner sind stets atomar und enden auf „=", Beispiele sind object-spec=, time=, feature=. Von den Werten, die ein Argument annehmen kann, gibt es verschiedene Arten. Die *einfachen Werte* sind atomar, z. B. room oder 42. Darüber hinaus gibt es verschiedene Arten *komplexer Werte*. Die erste Möglichkeit zur Bildung komplexer Werte ist der *Rahmen* (engl. *Frame*), eine Sammlung von atomaren Werten und Unterargumenten, zum Beispiel

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

In diesem Konstrukt bezeichnet man quantity= und for-whom= als Unterargumente von room-spec=. Die Spezifikation gibt zu jedem Argument a eine Menge von Werten und Argumenten an, die gültige *Köpfe* (engl. *Head* oder *Value*, aber nicht zu verwechseln mit „Wert") von a sind. Weiter gibt sie eine Menge von *Relationsargumenten* (engl. *Relation*) und von *Attributargumenten* (engl. *Attribute*) an. Ein Rahmen besteht nun aus genau einem Kopf und einer Reihe von Relations- und Attributargumenten, denen ihrerseits einfache oder komplexe Werte zugeordnet sind. Die Reihenfolge innerhalb eines Rahmens ist ohne Bedeutung. Ein Unterargument darf pro Rahmen höchstens einmal auftreten. Falls für einen Rahmen mehrere Köpfe oder eine mehrmalige Verwendung des gleichen Arguments nötig werden sollten, so ist meist der Einsatz von Listen (s. u.) erforderlich.

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 Attributsargumenten (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 quantity= sowohl die natürlichen Zahlen als auch das Unterargument percentage= als Kopf nennt.

Die zweite Art komplexer Werte sind *Listen* (engl. *sets*) von Werten, mit denen alle Arten 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 *1, 2, 3, 4, 5*, oder

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

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

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

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

Anders als in Rahmen, bei denen die Reihenfolge, in der Werte und Unterargumente angegeben sind, bedeutungslos ist, so daß also $a = (b = c, d = e)$ semantisch äquivalent zu $a = (d = e, b = c)$ ist, ist die Reihenfolge der Elemente in einer Liste im allgemeinen bedeutungstragend.

Die Spezifikation stellt insgesamt etwa 300 Argumentbezeichner zur Verfügung, zu den am häufigsten auftretenden gehören identifiability= (zur Unterscheidung von bestimmten und unbestimmten Artikeln), who=, quantity= und object-spec=. Die Möglichkeiten zur Bildung von Werten sind unbeschränkt. Zum einen sind auch Eigennamen und Zahlen als Werte möglich, so daß bereits die Zahl der atomaren Werte über alle Schranken wächst. Zum anderen impliziert die Möglichkeit der rekursiven Bildung von Werten aus anderen die Möglichkeit, beliebig tief verschachtelte Ausdrücke zu bilden: *... in dem Bett im Raum des Hotels auf dem Berg...*

$$\begin{aligned}\ldots \quad & \text{location} = (\text{bed}, \text{identifiability} = \text{yes}, \\ & \quad \text{location} = (\text{room}, \text{identifiability} = \text{yes}, \\ & \quad\quad \text{location} = (\text{hotel}, \text{identifiability} = \text{yes}, \\ & \quad\quad\quad \text{location} = (\text{mountain}, \text{identifiability} = \text{yes}, \ldots)))).\end{aligned}$$

3.3 Sprachübersetzung im Nespole! Projekt

Das Projekt Nespole! (**Ne**gotiating through **Spo**ken **L**anguage in **e**-Commerce) beschäftigt sich mit der Entwicklung einer Softwarearchitektur, die die Kommunikation zwischen Kunden und Anbietern elektronischer Dienste in anspruchsvoller Weise unterstützen soll. Neben multimedialen Elementen spielt dabei eine Komponente zur Übersetzung gesprochener Sprache eine zentrale Rolle. Die derzeit am Projekt beteiligten Sprachen sind Englisch, Deutsch, Französisch und Italienisch. Zwischen diesen wird mit Hilfe des Interchange Formats Übersetzung betrieben.

Die Architektur eines Prototyps des Gesamtsystems beschreiben Lavie *et al.* (2001). Hier interessieren nur die direkt an der Übersetzung beteiligten Komponenten, die in Abbildung 6 in einem vereinfachten Schema dargestellt sind.

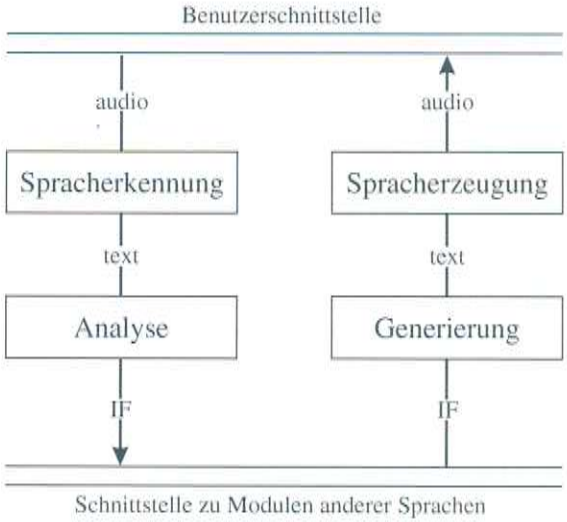

Abbildung 6 Vereinfachte Darstellung der Komponenten, die an der Sprachübersetzung beteiligt sind, nach Lavie *et al.* (2001)

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

Die Analyse geschieht in drei Schritten. Im ersten Schritt werden mit Hilfe handgeschriebener Grammatiken über Fragmenten der natürlichsprachlichen Eingaben Syntaxbäume erzeugt. Die Wörter der Eingabe werden dabei derart in Phrasen aufgeteilt, daß von den Syntaxbäumen ein möglichst großer Teil der Eingabe abgedeckt wird. Die Grammatiken, nach denen die Syntaxbäume erstellt werden, sind dabei so konzipiert, daß sich durch Löschen technischer Knoten aus den Syntaxbäumen Argumentbäume des Interchange Formats extrahieren lassen. Ein Beispiel für einen Syntaxbaum zeigt Abbildung 7.

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]

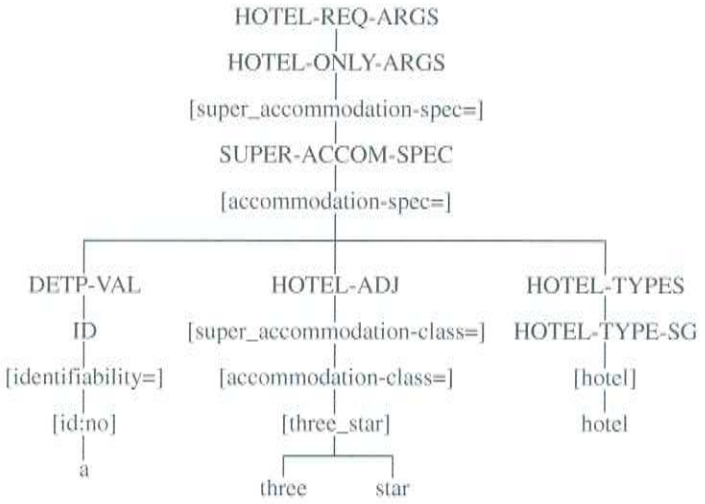

Abbildung 7 Syntaxbaum über einer englischen Wortfolge. Aus den entsprechenden Nonterminalen dieses Syntaxbaums wird der IF-Unterbaum accommodation-spec = (hotel, identifiability = no, accommodation-class = three-star) generiert. Entnommen aus (Han, 2002).

[7] In der Evaluation ergibt sich hieraus 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 Termsprache zu formulieren, sind nach Gleichung (1) auf Seite 20 ein Sprachmodell zur Schätzung von $p(\mathbf{e})$ ($\mathbf{e}$ nun ein Term) sowie ein Übersetzungsmodell zur Schätzung von $p(\mathbf{f}|\mathbf{e})$ ($\mathbf{e}$ ein Term und $\mathbf{f}$ 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 $\mathbf{s}$ ein Satz und w eine Wortform, so ist auch $(\mathbf{s}, w)$ 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 $\mathbf{t}_1, \ldots, \mathbf{t}_n$ Terme und ist f ein Symbol, so ist $\mathbf{t} = f(\mathbf{t}_1, \ldots, \mathbf{t}_n)$ ein Term. In diesem Fall heißt f die *Wurzel* des Terms $\mathbf{t}$ und n heißt die *Stelligkeit* (oder *Breite*) des Terms $\mathbf{t}$.[8] Die Terme $\mathbf{t}_1, \ldots, \mathbf{t}_n$ heißen *Unterterme* von $\mathbf{t}$.

[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 Terme aufzufassen sein werden: Dem Satz (w_1, w_2, w_3, w_4) etwa entspricht der Term $w_1(w_2(w_3(w_4)))$.

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

Als *Ebenen* in einem Term $\mathbf{t}$ bezeichnen wir die Menge aller Knoten gleicher Tiefe. Genauer heißt

$$L_\uparrow(i,\mathbf{t}) := \{s : \partial_\uparrow(s,\mathbf{t}) = i\}$$

die *Bottom-Up-Ebene* von $\mathbf{t}$ zum Index i und

$$L_\downarrow(i,\mathbf{t}) := \{s : \partial_\downarrow(s,\mathbf{t}) = i\}$$

die *Top-Down-Ebene* von $\mathbf{t}$ zum Index i.

Schließlich ist der i-te *Kopf* eines Terms $\mathbf{t}$ derjenige Term $\mathbf{t}' = \mathrm{top}_i(\mathbf{t})$, der aus $\mathbf{t}$ durch Ersetzung aller Symbole aus $L_\downarrow(i+1,\mathbf{t})$ samt zugehöriger Unterterme durch verschiedene Variablen entsteht.

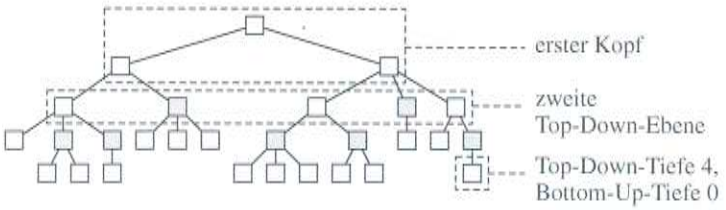

Abbildung 8 Größen in einem Baum. Die Menge der grauen Knoten bilden zusammen die Bottom-Up-Ebene 1.

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 erzeugten Wörter am wahrscheinlichsten ist:

$$\begin{aligned} p(\mathbf{e}) &= p(e_1,\ldots,e_l) \\ &= p(e_l|e_1,\ldots,e_{l-1}) \cdot p(e_1,\ldots,e_{l-1}) \\ &= \ldots \\ &= \prod_{i=1}^{l} p(e_i|e_1,\ldots,e_{i-1}). \end{aligned} \tag{8}$$

Der Generierung von Termen soll analog hierzu die Vorstellung zugrunde gelegt werden, daß jeder Teilterm dadurch generiert wird, daß zunächst rekursiv seine direkten Unterterme bestimmt werden, bevor unter Berücksichtigung dieser die Wurzel des Teilterms bestimmt wird. Dies führt für einen Term $\mathbf{t}$ mit Wurzel f und direkten Untertermen $\mathbf{t}_1,\ldots,\mathbf{t}_r$ in direkter Entsprechung zu Gleichung (8) auf die Wahrscheinlichkeitszerlegung

$$p(\mathbf{t}) = p(f|\mathbf{t}_1,\ldots,\mathbf{t}_r)\cdot\prod_{k=1}^{r} p(\mathbf{t}_k|\mathbf{t}_1,\ldots,\mathbf{t}_{k-1}).$$

Die Tatsache, daß in der speziellen Termsprache, die wir betrachten (Abschnitt 3.2), die Reihenfolge der direkten Unterterme 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 $p(\mathbf{t}_k|\mathbf{t}_1,\ldots,\mathbf{t}_{k-1})$ unabhängig von den $\mathbf{t}_1,\ldots,\mathbf{t}_{k-1}$ sind.

Diese Annahme führt auf die Form

$$p(\mathbf{t}) = p(f|\mathbf{t}_1,\ldots,\mathbf{t}_r)\cdot\prod_{k=1}^{r} p(\mathbf{t}_k), \tag{9}$$

in der die $p(\mathbf{t}_k)$ rechts sich nunmehr nach der gleichen Formel rekursiv weiter zerlegen lassen. Zu schätzen bleibt damit nur noch die Größe $p(f|\mathbf{t}_1,\ldots,\mathbf{t}_r)$. 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:

$$p(f|\mathbf{t}_1,\ldots,\mathbf{t}_r) \approx \frac{n(\,f(\mathrm{top}_{n-1}(\mathbf{t}_1),\ldots,\mathrm{top}_{n-1}(\mathbf{t}_r))\,)}{\sum_v n(\,v(\mathrm{top}_{n-1}(\mathbf{t}_1),\ldots,\mathrm{top}_{n-1}(\mathbf{t}_r))\,)}.$$

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 wirkungsvolles Hilfsmittel zur Beseitigung 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 $\mathbf{t}_1,\mathbf{t}_2$ genau dann die selbe Normalform $\mathbf{t}^*$ zuordnet, wenn sich $\mathbf{t}_1$ und $\mathbf{t}_2$ durch Permutation verschwisterter Unterbäume ineinander überführen lassen, ist schnell gefunden: Für jede beliebige *Termordnung* $\prec$ bildet zum Beispiel der Regelsatz

$$f(\mathbf{t}_1,\ldots,\mathbf{t}_i,\mathbf{t}_{i+1},\ldots,\mathbf{t}_r) \rightsquigarrow f(\mathbf{t}_1,\ldots,\mathbf{t}_{i+1},\mathbf{t}_i,\ldots,\mathbf{t}_r) \qquad \text{falls } \mathbf{t}_{i+1} \prec \mathbf{t}_i$$

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 komplizierterer Regeln wie

$$\text{operator} = x,[y = z_1, y = z_2, \ldots, y = z_n] \quad \rightsquigarrow \quad y = (\text{operator} = x,[z_1,\ldots,z_n])$$

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 $p(\mathbf{t}_k)$ als Produkt mehrerer Wahrscheinlichkeiten im allgemeinen weit kleiner sein wird als die Standardabweichung in der Verteilung von $p(f|\mathbf{t}_1,\ldots,\mathbf{t}_k)$, bevorzugt 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 Untertermwahrscheinlichkeiten zu verwenden. Dies führt auf das *Minimummodell* mit der Gleichung

$$p(\mathbf{t}) = p(f|\mathbf{t}_1,\ldots,\mathbf{t}_r)\cdot\min_{k=1}^{r} p(\mathbf{t}_k). \tag{10}$$

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 $\sum_{\mathbf{t}\in T} p(\mathbf{t}) < 1$. 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.

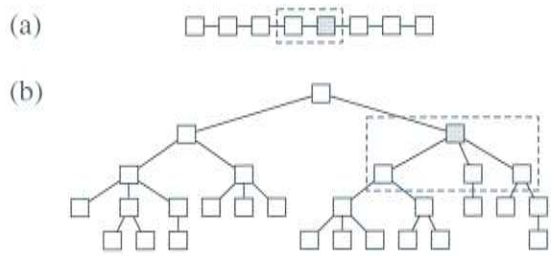

Abbildung 9 Das Konzept des n-Gramms für Sequenzen (a) und Bäume (b)

4.3 Übersetzungsmodelle für Termsprachen

Solange ein Übersetzungsmodell nur auf Übersetzungswahrscheinlichkeiten 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 $p(\mathbf{f}|\mathbf{t})$ ($\mathbf{f}$ sequentiell, $\mathbf{t}$ ein Term) durch eine beliebige Indizierung der Knoten von $\mathbf{t}$. Ist etwa $e_1,\ldots,e_l$ eine solche Indizierung, so überträgt sich das IBM-1-Modell wörtlich:

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

Höhere Modelle, in denen z. B. wie im IBM-2-Modell auch Zuordnungswahrscheinlichkeiten berücksichtigt werden, werfen dagegen die Frage nach einer adäquaten 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 $\mathbf{t}$ der „Zugriffspfad" von der Wurzel von $\mathbf{t}$ 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 $\mathbf{t}$. An die Stelle der Länge l von $\mathbf{t}$ träte die Äquivalenzklasse der Terme, deren Indexmenge (Menge der Indizes der Knoten eines Terms) gleich ist der Indexmenge von $\mathbf{t}$.

Da die Zahl der Äquivalenzklassen mit der Tiefe der Terme exponentiell steigt, ist nicht zu erwarten, daß zum Training der Wahrscheinlichkeiten für ein derartiges Modell hinreichend viel Material zur Verfügung steht. Zwar gehören die 3 885 Terme in unserer Trainingsmenge A (vgl. S. 79) nur 372 verschiedenen Äquivalenzklassen an, doch findet sich zu 243 dieser Klassen im gesamten Corpus nur je ein einziger Repräsentant. Eine Klassifizierung der Terme nach deren Indexmengen hätte demnach zur Folge, daß sich fast alle Terme auf wenige Klassen konzentrieren. Eine genauere Betrachtung zeigt, daß gerade die trivialen Terme riesige Klassen bilden, während mit zunehmender Komplexität der Termstruktur die Zahl der Terme pro Klasse rapide abfällt. Da aber gerade die komplizierteren Strukturen für das Training von Interesse sind (Übersetzungen von „ja", „nein" oder „hallo" bereiten ohnehin keine Schwierigkeiten), sind heuristische Annahmen unverzichtbar.

Motiviert durch die Kommutativität im Interchange Format, die sich bereits im Sprachmodell (s. o.) 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

$$p(\mathbf{f}|\mathbf{t}) = \prod_{j=1}^{m}\sum_{i=1}^{\partial\mathbf{t}} \frac{a(i|j,m,\partial\mathbf{t})}{|L_{\uparrow}(\mathbf{t},i)|}\sum_{e\in L_{\uparrow}(\mathbf{t},i)} t(f_j|e), \tag{11}$$

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 $a(i|j,m,\partial\mathbf{t})$ ausdrücken, mit welcher Wahrscheinlichkeit das j-te Wort eines Satzes mit m Wörtern in einem Term $\mathbf{t}$ einem Symbol auf Ebene i zugeordnet wird, drückt der zusätzliche Faktor $\frac{1}{|L_{\uparrow}(\mathbf{t},i)|}$ aus, daß die Zuordnung mit gleicher Wahrscheinlichkeit zu jedem Symbol auf dieser Ebene geschehen kann.

Die Verwendung von $L_{\uparrow}(\mathbf{t},i)$ hat gegenüber $L_{\downarrow}(\mathbf{t},i)$ den Vorteil, daß gleiche Unterterme vom Modell gleich behandelt werden unabhängig davon, wo sie sich befinden. Zum Beispiel würden bei Verwendung von $L_{\downarrow}(\mathbf{t},i)$ im Fall $\mathbf{t} = f(g(h,i),k(g(h,i)))$ 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 $(\mathbf{f},\mathbf{t})$ die Wahrscheinlichkeit $p(\mathbf{f}|\mathbf{t})$ maximieren, unter den Nebenbedingungen, daß $\sum_e t(f|e) = 1$ und $\sum_i a(i|j,m,d) = 1$ gelten. Gesucht sind also die Stellen, an denen die partielle Ableitung von

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

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

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

$$\Longleftrightarrow \qquad t(f|e) = \lambda_e^{-1} p(\mathbf{f}|\mathbf{t}) \cdot \underbrace{t(f|e) \sum_{j=1}^{m} \delta(f,f_j) \frac{\sum_{i=0}^{d} \frac{a(i|j,m,d)}{|L_i|} \sum_{e_i \in L_i} \delta(e,e_i)}{\sum_{i=0}^{d} \frac{a(i|j,m,d)}{|L_i|} \sum_{e_i \in L_i} t(f|e_i)}}_{=:c(f|e,\mathbf{f},\mathbf{t})}$$

und für $a(i|j,m,d)$ auf

$$\frac{\partial}{\partial a(i|j,m,d)} h(t,a,\lambda,\mu) = p(\mathbf{f}|\mathbf{t}) \cdot \sum_{j=1}^{m} \frac{\frac{1}{|L_i|} \sum_{e_i \in L_i} t(f_j|e_i)}{\sum_{k=0}^{d} \frac{a(i|j,m,d)}{|L_k|} \sum_{e_k \in L_k} t(f_j|e_k)} - \mu_{jmd} = 0$$

$$\Longleftrightarrow \qquad a(i|j,m,d) = \mu_{jmd}^{-1} p(\mathbf{f}|\mathbf{t}) \cdot \underbrace{\frac{1}{|L_i|} \sum_{j=1}^{m} \frac{\sum_{e_i \in L_i} t(f_j|e_i)}{\sum_{k=0}^{d} \frac{a(k|j,m,d)}{|L_k|} \sum_{e_k \in L_k} t(f_j|e_k)}}_{=:c(i|j,m,d,\mathbf{f},\mathbf{t})}$$

Mit diesen angepaßten Counts haben die Trainingsgleichungen die gewohnte Form

$$t(f|e) = \lambda'_e \sum_{(\mathbf{f},\mathbf{t}) \in C} c(f|e,\mathbf{f},\mathbf{t}), \qquad a(i|j,m,d) = \mu'_{jmd} \sum_{(\mathbf{f},\mathbf{t}) \in C} c(i|j,m,d,\mathbf{f},\mathbf{t}).$$

Die Konstanten λ'_e und μ'_{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

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 $a(i|j,m,\partial\mathbf{t})$ nicht von j und m abhängt:

$$p(\mathbf{f}|\mathbf{t}) = \prod_{j=1}^{m} \sum_{i=1}^{\partial\mathbf{t}} \frac{a(i|\partial\mathbf{t})}{|L_{\uparrow}(\mathbf{t},i)|} \sum_{e \in L_{\uparrow}(\mathbf{t},i)} t(f_j|e). \tag{12}$$

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

4.4 Decodierung

Die Generierung von Bäumen als Übersetzungen von Sätzen erfolgt analog zum Vorgehen im sequentiellen Fall. Dort (vgl. Abschnitt 2.4) werden beginnend mit der leeren Hypothese iterativ neue Hypothesen dadurch gewonnen, daß jeweils *eine* bestehende Hypothese um ein neues Wort ergänzt wird. Um Bäume zu generieren genügt es, dieses Verfahren dahingehend zu modifizieren, daß man neue Hypothesen durch Ergänzung *einer Menge* von bestehenden Hypothesen um ein neues Symbol erhält. Auf diese Weise werden von den Blättern ausgehend Bäume generiert, die in jedem Schritt zu Bäumen der nächstgrößeren Tiefe kombiniert werden. Dieses Verfahren ist in Abbildung 10 schematisch dargestellt (vgl. auch Abb. 5 auf Seite 29).

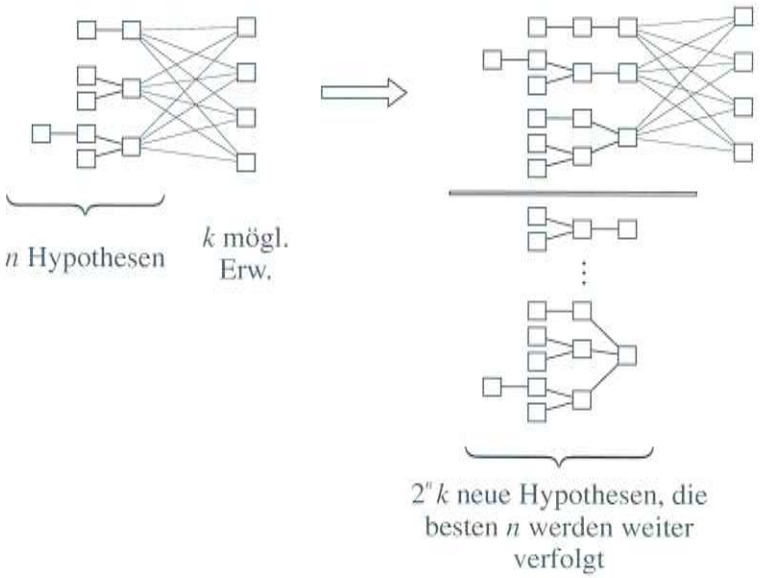

Abbildung 10 Analogon zu Abbildung 5: Decodierung von Bäumen durch Wahl einer Menge von Unterbäumen und eines neuen Symbols als Wurzel

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 überdies dadurch verschärft, daß es zur Generierung nicht-vollständiger Bäume[9] nötig 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 Termsprache der Suchraum erheblich einschränken (s. u.). Unter anderem dadruch gelingt es, die Explosion des Suchraums relativ weit hinauszuzögern.

Bei der *Bewertung* der Hypothesen unterscheiden sich die Präfixe des sequentiellen Falls von den Unterbäumen 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 späteren Gesamtwahrscheinlichkeit leisten (Abb. 11). Zwar mögen spätere benachbarte Unterbäume zum Zeitpunkt der Bewertung bereits decodiert sein und sich in der Menge der Hypothesen befinden, doch ist zum Zeitpunkt der Bewertung offen, welche Unterbäume zu einem späteren Zeitpunkt benachbart sein werden.

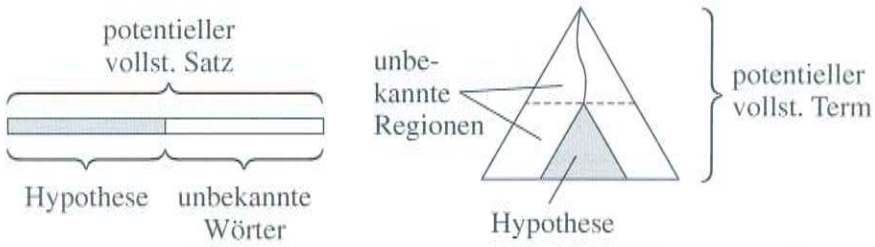

Abbildung 11 Bewertung einer Hypothese: Während im sequentiellen Fall nur der Beitrag der noch nicht decodierten Wörter abzuschätzen ist, sind bei der Bewertung eines decodierten Unterbaums zusätzlich die Beiträge der Nachbarregionen abzuschätzen.

Die Bewertungsfunktion τ 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:

$$\tau(H) := p(H) \cdot \sum_{d=\partial H}^{\infty} p(d|m) \prod_{j=1}^{m} \sum_{i=0}^{d} a(i|j,m,d)\,\tau(i,j,H). \tag{13}$$

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 $|L_\uparrow(H,i)|_+ := \max\{1, |L_\uparrow(H,i)|\}$ sei

$$\tau(i,j,H) := \sum_{b=|L_\uparrow(H,i)|_+}^{\infty} \frac{p(b|i,d)}{b} \left[\sum_{e_i \in L_\uparrow(H,i)} t(f_j|e_i) + (b - |L_\uparrow(H,i)|) \sum_{v} p(v)\,p(f_j|v) \right].$$

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

[9] gemeint sind damit Bäume, deren Blätter verschiedene Top-Down-Tiefen haben dürfen

Der normierende Faktor $1/b$ drückt die Gleichverteilung der Zuordnungswahrscheinlichkeiten innerhalb der i-ten Ebene aus und entspricht dem Faktor $1/|L_{\uparrow}(\mathbf{t},i)|$ 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_i tragen jeweils $t(f_j|e_i)$ bei, jedes der übrigen $b-|L_{\uparrow}(H,i)|$ noch unbekannten Symbole liefert einen Beitrag von $\sum_v p(v)p(f_j|v)$.

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 $|L_{\uparrow}(H,i)|=1$ genau dann, wenn das i-te Wort e_i bereits decodiert ist, ansonsten ist $|L_{\uparrow}(H,i)|=0$. Da der Wert der eckigen Klammer also nur für den Fall $b=1$ relevant ist, verschwindet wegen $|L_{\uparrow}(H,i)|\in\{0,1\}$ genau einer der beiden Summanden, was auf Seite 29 durch die Fallunterscheidung ausgedrückt wird.

Wie bereits angeklungen, sind bei einer Implementierung des Decoders spezielle Vorkehrungen nötig, die den Suchraum in geeigneter Weise beschneiden. 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 $\sum_{j=1}^{m} t(e_i|f_j)$ $(i=1,\ldots,k)$ maximal wird. Die $t(e_k|f_j)$ entstammen einem umgekehrt trainierten IBM-1-Modell. Die Anzahl k wird in Abhängigkeit von der Länge m von $\mathbf{f}$ 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.[10] Innerhalb des Decoders wird daher $\{e_1,\ldots,e_k\}$ als Lexikon der Zielsprache verwendet.

Die maximale Stelligkeit der Terme wird auf vier, ihre maximale Tiefe auf drei beschränkt. Pro Argument und Termtiefe wird die Zahl der Hypothesen auf acht beschränkt. Hier wurden die maximalen Zahlen gewählt, bei denen der Laufzeitbedarf des Decodierens noch im Rahmen bleibt. Aufgrund der exponentiellen Komplexität des Algorithmus resultieren geringfügig andere Schranken bereits in erheblich anderem Laufzeitverhalten. Wie die Beschränkung auf ein lokales Lexikon birgt auch die Einschränkung auf maximale Breite und Tiefe sowie die Beschränkung der Zahl der Hypothesen die Gefahr von Suchfehlern. Die Evaluation (Kapitel 7) wird später zeigen, daß die Implementierung des Decoders trotz dieser Beschneidungen des Suchraums nur äußerst wenige Suchfehler macht. Da diese Fehler meist im Zusammenhang mit unbekannten Wörtern stehen, ist anzunehmen, daß die Investition in zusätzliche Rechenzeit durch Ausweitung der Beschränkungen nur unmerkliche Verbesserungen bringen wird.

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

[10] Bei den übrigen Symbolen handelt es sich meist um Teile von Spezialkonstruktionen, oder es befindet sich unter den $\mathbf{e}_i$ ein Symbol mit semantisch ähnlicher Funktion.

aufgezählt, die sich durch Permutation ihrer Elemente ineinander überführen lassen. Nicht allgemein durch Iteration lösbar 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

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

dargestellt und kann in dieser Form weitgehend wie ein gewöhnlicher Term behandelt werden. Zu beachten ist lediglich, daß die Hilfsargumente *set=* und *empty=* für die Einschränkungen der Spezifikation „unsichtbar" sind. So ist im obigen Beispiel sicherzustellen, daß single von room-spec=, nicht etwa von *empty=* lizensiert wird. Zudem ist zu beachten, daß in einer Verwendung von *set=* mehrere Köpfe sowie mehrmalige Vorkommen von *empty=*-Untertermen gestattet sind. Das Argument *set=* darf an beliebigen Stellen auftreten, *empty=* jedoch nur direkt nach *set=*, und neben *empty=* dürfen einem *set=* nur atomare Wertbezeichner folgen.

Die Behandlung von Listen stellt während des Decodings ein besonderes Problem dar. Da hier Unterterme 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

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

generiert. Die maximale Anzahl von Werten pro Liste wird dabei durch eine geeignet gewählte Konstante beschränkt. Da die Werte, die in den Listen der neuen Hypothesen erscheinen, aus bestehenden Hypothesen zum gleichen Argument entnommen wurden, ist gewährleistet, daß keine nicht lizensierten Werte in einer Liste auftreten.

Neben den Listen stellt auch der Dialog Act eine spezielle Konstruktion im Interchange Format dar, die sich nicht unmittelbar auf allgemeine Termsprachen 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

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

ist zum Beispiel unklar, ob das Argument who= eher dem Konzept +reservation oder +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 Termsprache ausgehen.

Die starken Unterschiede zwischen der inneren Struktur des Dialog Acts und der des Argumentbaums motiviert eine Modularisierung, nach der jeder Ausdruck aus zwei Teilen besteht: Der *Wurzel* (den Dialog Act beinhaltend) und dem *Argumentbaum*. Bei Betrachtung des Argumentbaums tritt an die Stelle des Dialog Acts ähnlich wie bei der Behandlung von Listen das zusätzliche formale Argument *root*=. Dieses Argument lizensiert alle Unterargumente, die von irgendeinem Konzept oder Speech Act lizensiert werden, wird aber seinerseits von keinem anderen Argument lizensiert. Es definiert keine Köpfe und erlaubt keine atomaren Werte oder Listen. Argumentbäume dieser Form werden mit den oben beschriebenen Modellen und Algorithmen behandelt.

Die Wurzel hat für sich betrachtet eine sequentielle Struktur, die der der Konzeptketten bei Macherey *et al.* (2001) ähnelt. Es liegt daher nahe, zur Modellierung des Zusammenhangs zwischen natürlichsprachlichem Material und der Form der Wurzel nach dem Vorbild von Macherey *et al.* (2001) die Modelle für natürliche Sprachen unverändert zu übernehmen.

Die Decodierung eines vollständigen IF-Ausdrucks ist dann ein zweistufiger Prozeß. Zuerst wird wie in Abschnitt 4.4 beschrieben der Argumentbaum erzeugt. Der Baumdecoder übergibt die Liste der n besten Hypothesen an den Wurzeldecoder. 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 $\mathbf{t}_i$ ($i = 1, \ldots, n$) die decodierten Argumentbäume und ist $\mathbf{r}_i$ die zu $\mathbf{t}_i$ bestimmte Wurzel, so wird am Ende das Paar $(\mathbf{r}_i, \mathbf{t}_i)$ als Übersetzung von $\mathbf{f}$ zurückgegeben, für das das Produkt

$$\underbrace{p(\mathbf{r}_i) \cdot p(\mathbf{f}|\mathbf{r}_i)}_{\sim p(\mathbf{r}|\mathbf{f})} \cdot \underbrace{p(\mathbf{t}_i) \cdot p(\mathbf{f}|\mathbf{t}_i)}_{\sim p(\mathbf{t}|\mathbf{f})}$$

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.

Die aufwendige Zusammenstellung eines bilingualen Corpus, dessen eine Dimension Interlingua-Annotationen enthält, läßt sich durch *Projektion* bestehender Übersetzungsmodelle vermeiden. Dabei wird angenommen, daß Parameter für ein initiales Modell bekannt sind, etwa zur Übertragung von Englisch nach Interlingua. Unter Verwendung dieses Modells will man dann ein Modell zur Überführung von Deutsch in Interlingua durch „Zusammenschaltung" eines rein natürlichsprachlichen Modells für Deutsch/Englisch mit dem bestehenden für Englisch/Interlingua erhalten (Abb. 12). Ein derartiges Vorgehen erscheint lohnend, wenn der Verlust an Übersetzungsqualität gering ist im Vergleich zum ökonomischen Vorteil, den die Erstellung eines deutsch-englischen Corpus gegenüber einem Corpus für Deutsch/Interlingua bringt.

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 durch Projektion zweier Modelle F/G und G/E zu erhalten. Die Anwendung dieser Modelle auf die Übersetzungsmodelle aus Kapitel 4 stellt danach keine weitere Schwierigkeit dar.

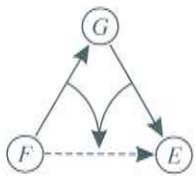

Abbildung 12 Schematische Darstellung der Projektion: Ein Übersetzungsmodell F/E zur Übersetzung von F nach E wird erzeugt durch Projektion eines Modells F/G und eines Modells G/E auf das Paar F/E.

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 verrauscht 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.

Die zweite potentielle Quelle für Verluste sind inhärente Differenzen in den beteiligten Sprachen. Wenn man sich Sprachen als Punkte in einem metrischen Raum vorstellt, in dem der Abstand $d(A,B)$ zwischen zwei Sprachen A und B ein Maß für die strukturellen Unterschiede zwischen diesen Sprachen darstellt, dann wird der potentielle Verlust durch den Umweg über eine dritte Sprache C gerade durch die Dreiecksungleichung beschrieben. Die zu erwartenden Probleme lassen sich an doppeldeutigen Wörtern wie dem deutschen *Schloß* veranschaulichen. Die Schließvorrichtung wird im Französischen mit *serrure* und im Englischen mit *lock* belegt; das Gebäude heißt im Französischen *château* und im Englischen *castle*. Ein direktes Training auf einem geeigneten französisch-englischen Corpus würde diese Paare so finden, doch für ein Modell F/E, das durch Projektion aus F/G und G/E gewonnen wurde, ist zu erwarten, daß *serrure* und *château* gleichermaßen in *lock* und *castle* übersetzt werden können. Zu dieser sogenannten *strukturellen Divergenz* vgl. auch Nasr *et al.* (1997).

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 Zuordnungswahrscheinlichkeiten $a(i|j,m,l)$ zu gewinnen. Wie üblich bezeichnen i,j Positionen und l,m die Längen von $\mathbf{e},\mathbf{f}$. Zusätzlich wird k als Position in $\mathbf{g}$ und n für die Länge von $\mathbf{g}$ verwendet. Statt t 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 Übergangsgraph 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

Verfügung stehen, erhält man die gesuchte Verteilung $p(f|e)$ also via

$$p(f|e) = \sum_g p(g)p(f|g)p(g|e). \tag{14}$$

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 $a(i|j,m,l)$ zu übertragen. Dieser Prozeß gestaltet sich verglichen 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 $p(l|n)$ wird zusätzlich die Zuordnungswahrscheinlichkeit $p(j|n,m)$ benötigt. Diese erhält man zum Beispiel, indem man parallel zum Training von $p(k|j,m,n)$ des Paars F/G ein vereinfachtes IBM-2-Modell für G/F trainiert, bei dem in $p(j|k,n,m)$ 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

$$p(m|l) = \sum_n p(n)p(m|n)p(n|l) \tag{15}$$

$$p(l|m) = \sum_n p(n)p(l|n)p(n|m). \tag{16}$$

Die $p(m|l)$ erlauben es dann, in einem zweiten Schritt die Hilfsverteilung $p(j|m,l)$ zu erzeugen. Wegen

$$\begin{aligned}
&\frac{1}{p(m|l)} \sum_n p(n)p(m|n)p(n|l)p(j|n,m) \\
&= \frac{p(l)}{p(m,l)} \sum_n \frac{p(n)p(m,n)p(n,l)p(j,m,n)}{p(n)p(l)p(n,m)} \\
&= \frac{1}{p(m,l)} \sum_n p(n,l)p(j,n,m) = \frac{p(j,m,l)}{p(m,l)} = p(j|m,l)
\end{aligned}$$

erhält man die $p(j|m,l)$ offenbar über

$$p(j|m,l) = \frac{1}{p(m|l)} \sum_n p(n)p(m|n)p(n|l)p(j|m,n). \tag{17}$$

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-

lung $p(i|j,m,l)$. Die Rechnung

$$\begin{aligned}
&\frac{1}{p(l|m)p(j|m,l)} \sum_n \Big(p(n)p(n|m)p(j|m,n)p(l|n) \cdot \\
&\qquad\qquad \sum_k p(k|j,m,n)p(k|n,l)p(i|k,n,l) \Big) \\
&= \frac{1}{p(l|m)p(j|m,l)} \sum_n \Big(\frac{p(n)p(m,n)p(j,m,n)p(l,n)}{p(m)p(m,n)p(n)} \cdot \\
&\qquad\qquad \sum_k \frac{p(k,j,m,n)p(k,n,l)p(i,k,n,l)}{p(j,m,n)p(n,l)p(k,n,l)} \Big) \\
&= \frac{1}{p(m)p(l|m)p(j|m,l)} \sum_n \sum_k p(k,j,m,n)p(i,k,n,l) \\
&= \frac{1}{p(j,m,l)} \sum_n p(i,j,m,n,l) = \frac{p(i,j,m,l)}{p(j,m,l)} = p(i|j,m,l)
\end{aligned}$$

führt auf die Projektionsformel

$$\begin{aligned}
p(i|j,m,l) = \frac{1}{p(l|m)p(j|m,l)} \sum_n \Big(& p(n)p(n|m)p(j|m,n)p(l|n) \cdot \\
& \sum_k p(k|j,m,n)p(k|n,l)p(i|k,n,l) \Big). \qquad (18)
\end{aligned}$$

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 Projektionsverfahren iteriert anwenden läßt, z. B. zur Überbrükkung mehrerer Sprachen (Abb. 13).

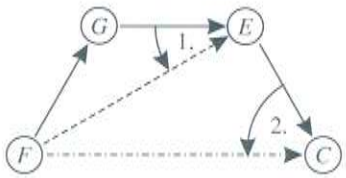

Abbildung 13 Doppelte Anwendung der Projektion, um ein Modell F/C (Französisch nach Chinesisch) aus drei Modellen F/G, G/E und E/C zu erzeugen.

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 minimalem Aufwand neue Ideen zu Forschungszwecken zu implementieren. Laufzeit- und Speichereffizienz spielten dagegen eine untergeordnete Rolle.

6.1 Grundkonzept des Systems

Als Benutzerschnittstelle dient ein Befehlsprompt, das es dem Benutzer erlaubt, in Java-ähnlicher Syntax Java-Objekte des Laufzeitsystems zu manipulieren. Ein etwaiger Rückgabewert einer ausgeführten Befehlszeile wird auf der Standardausgabe ausgegeben.

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

Neben den Datentypen boolean, int, long, double und String sind von der Benutzerschnittstelle alle Unterklassen der Klasse *Concept* sichtbar. Alle öffentlichen Methoden, die zu einem bestimmten Objekt deklariert sind, und die nur von zugänglichen Datentypen abhängen, können vom Benutzer auf den Objekten ausgeführt werden. Statt dem Punkt dient dabei der Doppelpunkt als Navigationssymbol. Sollten für einen bestimmten Methodenaufruf Typumwandlungen nötig werden, so werden diese implizit ausgeführt, wann immer dies möglich ist.

Neben den Methoden stellt das System einige globale *Funktionen* zur Verfügung. Zu ihnen gehört z. B. das im obigem 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.

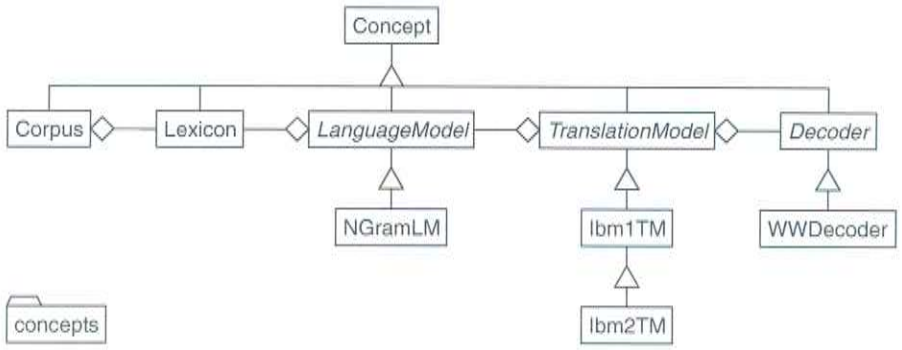

Abbildung 14 UML-Klassendiagramm zu einigen zentralen Unterklassen von *Concept*. Zur Übersichtlichkeit sind keine Methoden angegeben. Zu UML siehe (Balzert, 2001). Vgl. auch Abb. 28, S. 97.

Objekte der Klasse *Lexicon* dienen zur Umwandlung von Zeichenketten in interne Repräsentationen. Zur Verarbeitung einzelner Wortformen werden intern Ganzzahlen verwendet, die Methoden *string2int*() und *int2string*() dienen zur Konversion zwischen den Darstellungen. Ein *Lexicon* wird definiert durch Angabe eines *Corpus* (s. u.), die Zuordnung von Ganzzahlen zu Wortformen geschieht in lexikalischer Reihenfolge. Alle Zeichenketten, die das *Corpus* nicht enthielt, werden wie das spezielle Wort ⟨*unk*⟩ behandelt. Zur Repräsentation ganzer Sätze dient im System die Klasse *Phrase*, auch sie wird vom *Lexicon* unterstützt. Die Methoden *string2phrase*() und *phrase2string*() dienen zur Konvertierung zwischen Zeichenketten und *Phrase* Objekten. Das *Lexicon* kann dabei wahlweise normalisierend auf eine Zeichenkette einwirken, etwa um Interpunktionszeichen von eigentlichen Wörtern zu trennen oder um Groß- und Kleinschreibung außer Acht zu lassen.

Corpus-Objekte repräsentieren multilinguale 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 zurückliefert. Iteratoren für verschiedene Dimensionen durchlaufen dabei die Tupel in der gleichen Reihenfolge, so daß die i-ten Elemente verschiedener Iteratoren garantiert zum gleichen Tupel gehören. Corpora lassen sich via *add*() addieren (Konkatenation) und via *mult*() multiplizieren (kartesisches Produkt).

Die abstrakte Klasse *LanguageModel* definiert ein Sprachmodell. Objekte dieser Klasse dienen dazu, Phrasen und Wörtern einer bestimmten Sprache Wahrscheinlichkeiten zuzuordnen. Dazu muß ein Sprachmodell zunächst mit der Methode *learn*() auf eine Sprache trainiert werden. Als Trainingsmenge wird dabei eine Dimension eines *Corpus* angegeben. Falls das Modell noch nicht über ein *Lexicon* der Sprache verfügt, wird beim Anstoß des Trainings eines aus der Trainingsmenge erzeugt. Eine Implementierung von *LanguageModel* wird in der Klasse *NGramLM* bereitgestellt. Sie implementiert ein klassisches n-Gramm-Modell, wie es in Abschnitt 2.2 beschrieben ist.

Entsprechend dem *LanguageModel* gibt es eine abstrakte Klasse *TranslationModel* zur Definition von Übersetzungsmodellen. Es dient wie in der Theorie zur Schätzung der Wahrscheinlichkeit $p(\mathbf{f}|\mathbf{e})$ für einen Quellsatz $\mathbf{f}$ und einen Zielsatz $\mathbf{e}$. Weiter verfügt es über *LanguageModel*-Objekte für Quell- und Zielsprache sowie über einen *Decoder* zur Generierung von Übersetzungen. Zum Training stehen diverse *learn*()-Methoden zur Verfügung. Vollständig implementiert wird *TranslationModel* unter anderem von einer Klasse *Ibm1TM*, in der Training und die Bestimmung von $p(\mathbf{f}|\mathbf{e})$ nach dem IBM-1-Modell (Abschnitt 2.3) realisiert sind. Die Unterklasse *Ibm2TM* von *Ibm1TM* realisiert das IBM-2-Modell. Die beiden speziellen Klassen erlauben über Methoden $t()$ und $a()$ auch einen direkten Zugriff auf die Verteilungen $t(f|e)$ und – im IBM-2-Modell – $a(i|j,m,l)$. Ferner stellen sie Methoden zur expliziten Bestimmung von Zuordnungen zur Verfügung. Die Projektion von IBM-Modellen nach Kapitel 5 wird in der Implementierung als *Multiplikation* von Übersetzungsmodellen bezeichnet und für die IBM-Modelle von der Methode *mult*() implementiert. Die zusätzlich nötigen Wahrscheinlichkeitsverteilungen werden beim Training mitbestimmt.

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.

Schließlich wird die Definition einer Klasse *Decoder* zusammen mit einigen Implementierungen bereitgestellt. *Decoder* stellen im Wesentlichen eine Methode *trans*() zur Verfügung, die in Abhängigkeit von einem Übersetzungsmodell und einem Quellsatz (Klasse *Phrase*) einen Zielsatz generiert. Diese Methode wird von der *trans*()-Methode in *TranslationModel* auf dem aktuell definierten *Decoder* des Modells aufgerufen. Neben einigen experimentellen Decodern existiert die Klasse *WWDecoder*, die den Wang-Decoder aus Abschnitt 2.4 für ein IBM-2-Modell implementiert.

[11] An der Benutzerschnittstelle kann man auch das Infix-Symbol '+' zur Addition verwenden. Intern wird $a+b$ wie $a : add(b)$ behandelt.

Der Einsatz der beschriebenen Klassen soll nun an einer Beispielsitzung demonstriert werden. Dazu wird angenommen, daß die Datei `train.corp` ein zweidimensionales Trainingscorpus enthält, in dessen nullter Dimension Äußerungen der Quellsprache stehen, denen in der ersten Übersetzungen in die Zielsprache gegenübergestellt sind. Als Sprachmodell der Zielsprache wird ein Trigramm-Modell und als Übersetzungsmodell ein IBM-2-Modell verwendet. Vom Sprachmodell der Quellsprache wird nur das *Lexicon* benötigt, so daß ein triviales Modell hier genügt. Zunächst wird über fünf Iterationen ein IBM-1-Modell trainiert. Mit diesem Modell wird ein IBM-2-Modell initialisiert, das auch über fünf Iterationen trainiert wird. In dieser Konfiguration werden dann Quellsprachformulierungen einer Datei `test.in` mit Hilfe eines Wang-Decoders (vgl. Abschnitt 2.4) übersetzt. Die Übersetzungen werden in der Datei `test.out` abgelegt.

```
> 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 Beispielsitzungen 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.

Eine *IFPhrase* repräsentiert einen Ausdruck des Interchange Formats durch ein *Root*-Objekt für den Dialog Act und ein *Tree*-Objekt für den Argumentbaum. Während eine *Phrase* sonst üblicherweise von einem *Lexicon* erzeugt wird, dient zur Generierung von *IFPhrase*-Objekten ein *IFLexicon*. Neben der Konvertierung zwischen Zeichenketten und Ganzzahlen für einzelne Symbole bietet das *IFLexicon* zusätzlich zu jedem Symbol (mindestens) ein Objekt der Klasse *IFElement* an. Diese Objekte beinhalten die Spezifikation des Interchange Formats und geben Auskunft über Lizensierungen und dergleichen. Zu ihrer Erzeugung sind bei Konstruktion eines *IFLexicon* die nötigen Spezifikationsdateien anzugeben. Weiter bietet das *IFLexicon* Dienste zur Überprüfung der semantischen Korrektheit von gegebenen *IFPhrase*-Objekten.

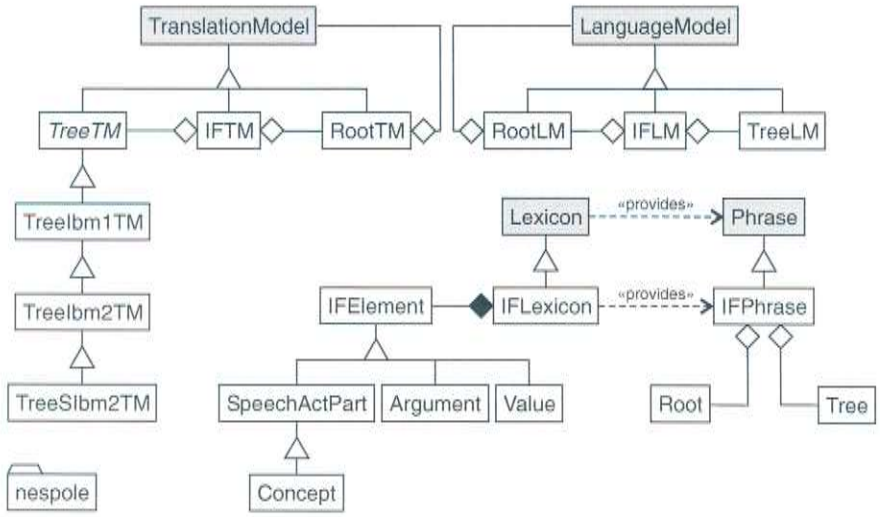

Abbildung 15 Die Klassen des Pakets `nespole`. Grau hinterlegte Klassen gehören zu anderen Paketen.

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

$$\text{c : give-information} + \text{disposition} + \text{trip}(\text{visit-spec} = \text{vacation}, \text{disposition} = \text{desire})$$

entgegenzunehmen 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

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

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 *TreeIbm2TM*, findet sich in einer Klasse *TreeIbm2Decoder*.

Die Verwendung der Klassen des Paketes `nespole` soll nun zum Schluß wieder an einer Beispielsitzung demonstriert werden. Wie zuvor beinhalte die Datei `train.corp` eine Trainingsmenge. Deren nullte Dimension enthalte Formulierungen einer natürlichen Sprache und die erste Dimension die zugehörigen Ausdrücke des Interchange Formats. Ebenfalls wie zuvor sei ferner `test.in` eine Menge zu übersetzender Formulierungen. Die dazu generierten Übersetzungen werden in der Datei `test.out` abgelegt. Zur Erzeugung des *IFLexicon* werden die offiziellen Spezifikationsdateien `da.db.lsp`, `nespole-arguments.db.lsp` und `nespole-values.db.lsp` benötigt (vgl. IF, 2002, und S. 97).

```
> 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.IFLM(tgt-root, tgt-tree)
> ibm1-root = new nespole.RootTM(new Ibm1TM(src, tgt-root))
> ibm1-root:learn(5, c, 0, 1)
> ibm2-root = new nespole.RootTM(new Ibm2TM(src, tgt-root))
> ibm2-root:learn(5, c, 0, 1, ibm1-root)
> ibm2-root:setRootDecoder(new nespole.RootIbm2Decoder())
> ibm1-tree = new nespole.TreeIbm1TM(src, tgt-tree)
> ibm1-tree:learn(5, c, 0, 1)
> ibm2-tree = new nespole.TreeIbm2TM(src, tgt-tree)
> ibm2-tree:learn(5, c, 0, 1, ibm1-tree)
> ibm2-tree:setTreeDecoder(new nespole.TreeIbm2Decoder())
> tm = new nespole.IFTM(ibm2-root, ibm2-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

Dieses Kapitel widmet sich der Evaluation der vorgestellten Verfahren. Ihre Ergebnisse werden sowohl mit den Ergebnissen menschlicher Experten als auch mit vergleichbaren Ergebnissen anderer Softwaresysteme verglichen. Als Corpora standen drei Mengen *A*, *B* und *C*, bestehend aus 3 885, 2 427 bzw. 194 Tripeln zur Verfügung, die jeweils äquivalente deutsche und englische Formulierungen sowie eine entsprechende Darstellung im Interchange Format umfassen. Die Anwendungssituation ist zumeist die Übersetzung von Deutsch ins Interchange Format, die englische Dimension wird in Abschnitt 7.4 zur Projektion verwendet.

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 ungesehen betrachtet werden. Tabelle 1 zeigt statistische Kenngrößen der drei Mengen, eine ausführliche Analyse der Corpora findet man im Anhang ab Seite 79.

Menge	*A*		*B*		*C*	
Sprache	D	E	D	E	D	E
Sätze	3 885	3 885	2 427	2 427	194	194
versch. Sätze	2 135	2 113	1 395	1 395	143	141
einm. Sätze	1 970	1 951	1 283	1 290	126	123
	50,70%	50,21%	52,86%	53,15%	64,94%	63,40%
Tokens	17 434	18 248	11 236	11 729	889	955
Wörter	1 558	1 213	1 196	1 010	269	241
einm. Wörter	755	489	566	429	152	123
	48,46%	40,30%	47,34%	42,47%	56,50%	51,01%

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 Satzpaares 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 zuordnen 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 Testsätzen zu den zugehörigen Argumentbäumen, die per Hand bestimmt wurden, zeigten, daß sich nur etwa 58,32% der deutschen Wörter sinnvoll Symbolen des Argumentbaums zuordnen ließen.[12] Betrachtet man die Zuordnungen von den gleichen deutschen Sätzen zu ihren englischen Übersetzungen, so kommt man auf eine Quote von 87,97%. 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

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

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.[13] 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

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

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_i mit

$$i = \arg\max_k t(f_j|e_k)$$

[12] Hier und im folgenden werden nur nichttriviale Paare betrachtet.

[13] 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 verursachen. Zum anderen kann man die Frage stellen, ob z. B. das *hier* im Beispiel wirklich semantiktragend ist, oder eher eine sprachinterne Funktion hat.

und beim zweiten Modell dem Symbol e_i mit

$$i = \arg\max_k a(\partial_\uparrow e_k | j, m, d) t(f_j | e_k)$$

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]

Obwohl diese Zahlen einen anderen Eindruck erwecken, lassen sich zum Teil qualitative Unterschiede zwischen den Zuordnungen der beiden Modelle erkennen. Da sich die Mehrzahl der Wörter Blättern eines Argumentbaumes zuordnen lassen, tendiert das IBM-2-Modell durch seine Zuordnungswahrscheinlichkeiten dazu, Zuordnungen zu Blättern zu bevorzugen, während das IBM-1-Modell bezüglich der Position im Baum keine Präferenzen hat. Dies führt dazu, daß es dem IBM-1-Modell eher gelingt, richtige Zuordnungen auf höheren Ebenen des Baums zu finden, wie der Vergleich in Abbildung 16 zeigt. Hier findet das IBM-1-Modell die Zuordnung von *klingt* zum Argument modifier=, das IBM-2-Modell nicht. Für die Funktionsweise des in Abschnitt 4.4 vorgestellten Decoders, der den Argumentbaum ausgehend von den Blättern von unten nach oben aufbaut, ist es jedoch von Vorteil, wenn im Zweifelsfall ein stärkeres Gewicht auf die Blätter gelegt wird. Eine Unsicherheit auf höherer Ebene kann ja im übrigen durch den Einfluß des Sprachmodells ausgeglichen werden, das bei der Wahl der Blätter nur einen geringen Einfluß ausüben kann.

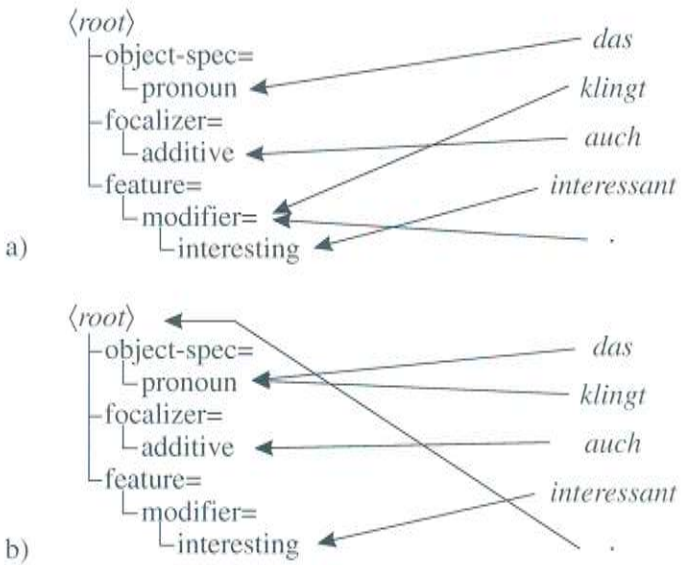

Abbildung 16 Zuordnung zwischen einem deutschen Satz und dem zugehörigen Argumentbaum a) nach dem IBM-1-Modell, b) nach dem IBM-2-Modell.

[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 qualitative 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.

Es wurde auch untersucht, wie gut die Zuordnungen zu den Elementen des Dialog Acts möglich sind, der ja durch ein eigenes Modell beschrieben wird. Zwar zeigt sich dabei, daß sich noch weit weniger Wörter sinnvoll eindeutig zu einem Konzeptbezeichner oder dem Speech Act zuordnen lassen als das beim Argumentbaum der Fall ist, nämlich nur etwa 26,46%, doch reicht dies im allgemeinen aus, da der Dialog Act ohnehin nur wenige Symbole umfaßt, die von einer Zuordnung getroffen werden müssen. Eine tiefergehende Darstellung der Zuordnungen von Wörtern zu Wurzelsymbolen erscheint im Hinblick auf die Qualität der erzeugten Wurzeln an dieser Stelle nicht nötig. Wie später zu sehen sein wird, erzeugen die Übersetzungsmodelle zusammen mit Sprachmodell und Decodierungsverfahren nämlich in der überragenden Zahl der Fälle Wurzeln, die mit der Wurzel des Referenzterms übereinstimmen.

	Deutsch/IF	Deutsch/Englisch
Optimale Quote r	58,32%	87,97%
IBM 1 Quote	$r \cdot 86{,}67\%$	$r \cdot 89{,}35\%$
IBM 2 Quote	$r \cdot 87{,}73\%$	$r \cdot 82{,}59\%$

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 grammatisch 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.

Zunächst der Vergleich zur Referenz der Datenbank. Ein Vergleich mit idealen Ergebnissen ermöglicht eine Unterscheidung zwischen zwei Fehlerarten eines statistischen Übersetzers: *Modellfehler* und *Suchfehler*. Ein Suchfehler liegt vor, wenn die Referenzübersetzung vom statistischen Modell besser bewertet wird als die gefundene Hypothese. Die bezüglich des Modells optimale Übersetzung lag dann in einem Teil des Suchraums, den der Decoder aus Effizienzgründen nicht betrachtet hat. Wenn dagegen die gefundene Hypothese besser bewertet wird als die Referenz, spricht man vom

Modellfehler, weil das Modell die Referenz zu schlecht bewertet. Modell- und Suchfehler schließen sich nicht gegenseitig aus, vielmehr ist davon auszugehen, daß bei jedem falschen Ergebnis beide Unsicherheiten eine Rolle spielen. Das Verhältnis der Bewertungen gibt allenfalls Auskunft über den Fehler, der am stärksten ins Gewicht fällt (Wang, 1998).

Bei etwa einem Drittel der Testmenge (32,99%) ist gar kein Fehler feststellbar, weil die Hypothese mit der Referenz deckungsgleich ist. Die voneinander abweichenden Übersetzungen zeigen eine ausgeprägte Tendenz zu Modellfehlern: 62,37% der Hypothesen werden vom Modell besser bewertet als die zugehörigen Referenzen, in nur 4,64% der Fälle handelt es sich um einen Suchfehler. Dies ist vor allem eine Aussage darüber, daß der Decoder in der Lage ist, den enormen Suchraum geeignet zu organisieren. (Man erinnere sich, daß beim Decodieren von Bäumen der Suchraum exponentiell größer ist als beim Decodieren von Sequenzen, vgl. Abschnitt 4.4).

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

$$\begin{aligned} &\mathrm{c} : \text{request-information} + \text{existence} + \text{price} \\ &\qquad (\text{price-spec} = (\text{for-whom} = (\text{quantity} = \text{plural}, \text{child}), \text{discount}), \\ &\qquad\ \text{location} = \text{there}) \end{aligned}$$

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

$$\begin{aligned} &\mathrm{c} : \text{request-information} + \text{existence} + \text{price} \\ &\qquad (\text{price-spec} = (\text{quantity} = \text{plural}, \text{discount}), \\ &\qquad\ \text{for-whom} = (\text{quantity} = \text{plural}, \text{child})), \end{aligned}$$

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 *bin ich da mit der ⟨unk⟩ verbunden* (g051ak.2.1, das fehlende Wort ist *Touristenauskunft*) wird zum Beispiel der Term

$$\mathrm{c} : \text{request-introduce-self}\ (\text{who} = \text{tourist\_information\_center}, \text{communication-mode} = \text{phone})$$

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

$$\mathrm{c} : \text{give-information} + \text{concept}\ (\text{concept-spec} = (\text{modifier} = (\text{nice}, \text{intensity} = \text{intense})))$$

liefert, 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.

Vergleich	besser	schlechter	ähnlich	gleich
IBM 2 vs. Grammatik, gesamt	13,92%	37,63%	48,45%	30,93%
IBM 2 vs. Grammatik, Wurzel	23,71%	24,23%	52,06%	42,27%
direkt trainiert vs. projiziert	27,32%	14,95%	57,75%	24,23%

Tabelle 3 Vergleich zwischen je zwei Arten Ausdrücke zu generieren. Angegeben ist jeweils der Anteil der Testmenge C, die vom erstgenannten System besser, schlechter oder ähnlich gut übersetzt wurde. Die letzte Spalte gibt an, wie viele Ausdrücke exakt übereinstimmen. Zur dritten Zeile vgl. Abschnitt 7.4

Die Grammatik ist unschlagbar, wenn es sich beim eingegebenen Satz um eine Konstruktion handelt, die von den Regeln der Grammatik abgedeckt wird. Aufgrund der Einfachheit des Corpus und des enormen Entwicklungsvorsprungs des grammatischen Systems trifft dies für einen großen Teil der Testmenge zu. Mit zunehmender Satzlänge

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

$$\begin{aligned} &\text{c : request-information} + \text{feasibility} + \text{view} + \text{object} \\ &\qquad (\text{location} = \text{question}, \\ &\qquad\ \text{object-spec} = (\text{whose} = (\text{child}, \text{quantity} = \text{plural}), \\ &\qquad\qquad\qquad\ \text{focalizer} = \text{additive}, \\ &\qquad\qquad\qquad\ \text{modifier} = \text{good})). \end{aligned}$$

Die Subjunktion *wo* wurde hier offensichtlich als Fragewort aufgefaßt (im Speech-Act request-information und im Unterterm 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

$$\begin{aligned} &\text{c : give-information} + \text{feature} + \text{activity} \\ &\qquad (\text{who} = (\text{focalizer} = \text{additive}, \text{child}), \\ &\qquad\ \text{activity-spec} = (\text{skiing}, \text{modifier} = (\text{intensity} = \text{medium}, \text{good}))). \end{aligned}$$

Es drückt in etwa aus, daß „auch Kinder" (who = ...) „ganz gut" (modifier = ...) „(Ski) fahren" (skiing). Aus Sicht des Argumentbaums ist dies nahezu ideal, ein Schönheitsfehler 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 statistischen 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 gleichauf 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

$$\text{c : give-information} + \text{feasibility} + \text{activity}$$

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

$$\text{c : give-information} + \text{existence} + \text{attraction}.$$

Statistisch bekommt man in diesen Fällen

$$\text{c : request-information} + \text{feasibility} + \text{activity}$$

bzw.

$$\text{c : negate-give-information} + \text{existence} + \text{object}.$$

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

$$\text{c : give-information} + \text{feature} + \underline{\text{object}}$$

ein besseres Ergebnis als das von der Statistik gelieferte c : 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

$$\text{c : introduce-topic} + \text{information-object},$$

verglichen mit dem allgemeineren c : 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

Neben einem direkten Vergleich auf der Ebene der erzeugten Terme ist auch eine Betrachtung der natürlichsprachlichen Übersetzungen von Interesse, die aus den gefundenen Termen generiert werden. Man spricht von einer *End-to-End-Evaluation*. Dazu werden die Terme, die mit Hilfe der grammatischen bzw. der statistischen Verfahren gewonnen wurden, von einem vorhandenen Generierer in englische Formulierungen umgewandelt und verglichen. Zusätzlich wurden die Ergebnisse eines rein statistisch arbeitenden Übersetzers in die Betrachtung einbezogen. Das rein statistische System verwendet Modelle, die von Vogel and Ney (2000) beschrieben wurden.

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

Zur Bewertung wurde jede der Übersetzungen von sechs Personen einer der Kategorien „perfect", „okay", „bad" zuzuordnen (vgl. auch das Bewertungsverfahren von Wang, 1998). Als perfekt sollte eine Übersetzung angesehen werden, die inhaltlich vollständig erfaßt und in der Zielsprache korrekt formuliert ist. Eine Übersetzung ist okay, wenn sie nicht perfect ist, aber wesentliche Teile des Inhalts verständlich formuliert werden konnten. Völlig unverständliche oder sinnverfälschende Übersetzungen sind bad. Beispielsweise wäre zum deutschen Satz *gibt es Unterkunft mit Küche* (Segment g047ak.8.2, vgl. S. 83f) die Übersetzung *Is there accommodation with kitchen* perfect, *Is there accommodation include kitchen to* okay und *There would be something for accommodation with ones* bad. Die Kategorie „acceptable" beinhaltet all jene Sätze, die perfect oder okay bewertet wurden.

Die Evaluation der Ergebnisse geschah parallel, d. h. während der Bewertung standen zu jedem Satz die Ausgaben der verschiedenen System nebeneinander, ähnlich der Darstellung im Anhang (vgl. S. 83). Dabei war den Bewertern nicht bekannt, welche Ausgabe von welchem System stammte. Die Bewertungen sind in Abbildung 17 dargestellt.

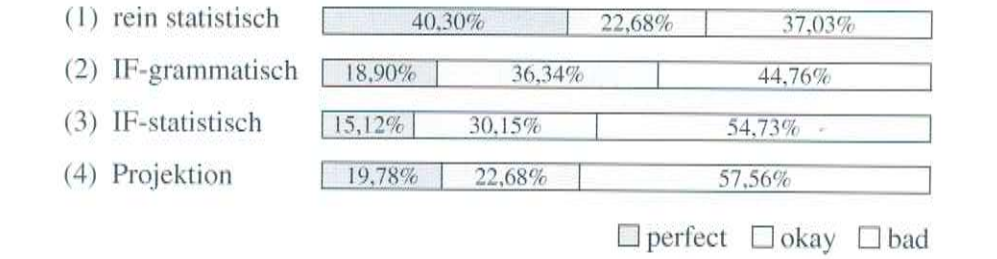

Abbildung 17 Durchschnittliche Bewertung der englischen Übersetzung durch verschiedene Evaluatoren. Angegeben sind jeweils die Anteile der Übersetzungen, die als perfekt (links), okay (mitte) und bad (rechts) bewertet wurden. Zu (4) vgl. Abschnitt 7.4.

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 verrauschter 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

$$\begin{aligned} &\mathrm{c} : \text{give-information} + \text{feature} + \text{activity} \\ &\qquad (\text{activity-spec} = \text{skiing}, \text{feature} = (\text{modifier} = \text{good})), \end{aligned}$$

der vom Generierer perfekt in *Skiing would be good.* übersetzt wird. Der statistische

Ansatz liefert mit

$$\begin{aligned} &\text{c : give-information} + \text{feature} \\ &\qquad (\text{activity-spec} = \text{skiing}, \text{feature} = (\text{modifier} = \text{good})), \end{aligned} \tag{19}$$

ein geringfügig anderes Ergebnis, doch mit diesem kommt der Generierer weit weniger 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

$$\begin{aligned} &\text{c : give-information} + \text{feature} + \text{activity} \\ &\qquad (\text{who} = (\text{focalizer} = \text{additive}, \text{child}), \\ &\qquad\ \text{activity-spec} = (\text{skiing}, \text{modifier} = (\text{intensity} = \text{medium}, \text{good}))), \end{aligned}$$

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 codiert ist. Man würde eher ein Ergebnis der Form *Children do skiing somewhat good, too.* erwarten.

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 verarbeitet 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 Ausdrücken, die von der Grammatik und der Statistik generiert werden, in der End-to-End-Betrachtung im wesentlichen erhalten bleibt. Die Übersetzungsqualität der interlinguabasierten statistischen Übersetzung liegt nur wenig unter der der etablierten Systeme, was bei Berücksichtigung des großen Entwicklungsvorsprungs dieser Systeme bereits als Nachweis für den Erfolg des statistischen Ansatzes gelten kann.

7.4 Verluste durch Projektion

Die Datenbank von Nespole!, aus der unsere Trainingsmengen extrahiert wurden, ist als multilinguales Corpus konzipiert. Jedes Element dieser Datenbank besteht aus einem Segment in der Originalsprache, einer Interchange-Format-Annotation, und optionalen Übersetzungen in andere Sprachen, die am Projekt teilnehmen. Die zwei dominanten natürlichen Sprachen sind Deutsch und Englisch, und unsere Trainings- und Testmengen wurden so gewählt, daß zu jedem Paar Deutsch/IF auch eine englische Übersetzung zur Verfügung steht.

Dies bietet eine ideale Voraussetzung für eine Evaluation der Projektionsmethode (Kapitel 5). Ein trainiertes Modell zur Übersetzung von Deutsch ins Interchange Format wird dabei verglichen mit einem Modell, das durch Projektion eines Modells für Deutsch-Englisch und eines Modells Englisch-IF erhalten wurde. Durch die Verwendung eines trilingualen Corpus anstelle zweier bilingualer Corpora werden bei einem solchen Vorgehen Verluste ausgeschlossen, die allein vom Abstand der beiden Datensätze herrühren.

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 ungenaue 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 Abstandsmaß 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.

Die Verhältnisse werden in Abbildung 18 dargestellt. Darin sind die drei betrachteten Sprachen in einem Dreieck ähnlich der allgemeinen Abbildung 12 dargestellt, wobei die Kantenlängen proportional zu den Logarithmen der Phrasenperplexitäten (vgl. S. 28) gewählt sind. Die Darstellung bestätigt zunächst die Vorstellung, daß der Abstand zwischen dem Interchange Format und einer natürlichen Sprache größer ist als der Abstand zwischen zwei natürlichen Sprachen. Ferner zeigt sie, daß die Perplexität des projizierten Modells Deutsch/IF nur geringfügig höher ist als die des direkt trainierten Modells Deutsch/IF. Der Logarithmus der Phrasenperplexität beträgt 57,96 beim trainierten Modell und 61.93 beim projizierten. Der Wert des projizierten Modells wird dabei insbesondere durch den Wert des trainierten Modells für Englisch/IF bestimmt, der bei 61,64 liegt, so daß der eigentliche Verlust durch die Projektion mit nur 0,29 Punkten verschwindend gering ist.

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 Terme 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 ihre Entsprechungen des projizierten Modells, in immerhin 14,95% der Fälle ist das Ergebnis des projizierten Modells besser.

Diese Zahlen spiegeln sich auch bei einer vergleichenden Humanevaluation wieder. Dort liegt der Anteil der akzeptablen Übersetzungen, die mit dem projizierten Modell (4) erzielt werden können, geringfügig unter dem entsprechenden Wert des direkt trainierten Modells (3), Abbildung 17. Überraschend ist dabei der Anteil der als perfekt bewerteten Übersetzungen. Dieser liegt mit 19,78% sogar noch über den 18,90% des grammatischen Systems. Der Umweg über eine zweite natürliche Sprache scheint in bestimmten Situationen glättend auf die Wahrscheinlichkeitsverteilungen zu wirken, so daß die resultierenden Übersetzungen weniger stark verrauscht sind und deshalb besser vom Generierer weiterverarbeitet werden können.

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

$$\begin{aligned} &\text{c : give-information} + \text{feature} + \text{object} \\ &\qquad (\text{who} = \text{pronoun}, \\ &\qquad\ \text{feature} = (\text{accommodation-spec} = (\text{accommodation-class} = \text{three\_star}))) \end{aligned}$$

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" bewertetet 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:

$$\begin{aligned} &\text{c : request-information} + \text{obligation} + \text{existence} + \text{object} \\ &\qquad (\text{property} = \text{already}, \text{experiencer} = \text{i}). \end{aligned}$$

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

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.

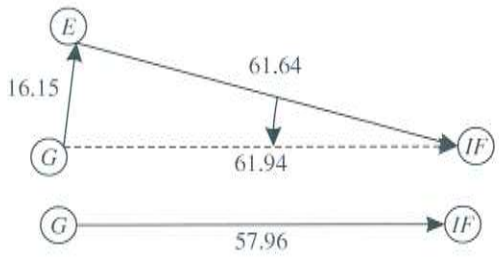

Abbildung 18 Abstände zwischen Englisch, Deutsch und Interchange Format gemessen in Logarithmen der Phrasenperplexitäten von IBM-2-Modellen.

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 Trainingmengen, die für die Untersuchungen dieser Arbeit zur Verfügung stehen, sind mit 2 427 bzw. 3 885 Paaren in der Tat nicht besonders groß,[16] 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.

Die unvermeidliche Verrauschtheit von Corpora führt dazu, daß die gesuchten Maxima in einem Corpus nicht exakt mit den „wahren" Maxima übereinstimmen. Dies kann dazu führen, daß man sich beim Training von Parametern zwar immer weiter dem Maximum der Trainingsmenge annähert, sich zugleich aber vom wahren Maximum wieder zu entfernen beginnt. Man spricht in dieser Situation vom *Übertrainieren*. Wenn man annimmt, daß das Rauschen in Trainingsmengen mit ihrer Größe abnimmt, daß also starkes Rauschen auf zu wenig Trainingsdaten hinweist, dann kann man Übertrainieren als Indikator dafür ansehen, daß das Potential der Modelle von der verwendeten Trainingsmenge nicht völlig ausgeschöpft werden kann.

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 ε-Umgebung um das Optimum des Corpus, die das wahre Maximum nicht mehr umfasst: Das Modell wird übertrainiert.

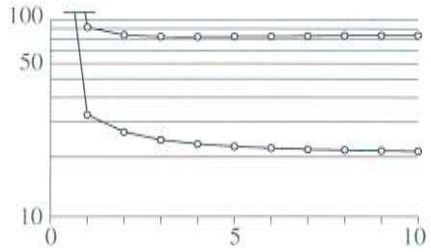

Abbildung 19 Vergleich der Perplexitäten von Trainings- und Testmenge eines Übersetzungsmodells für den Dialog Act. Die vertikale Achse ist logarithmisch.

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 Interchange 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 verschieden große Teilmengen der Menge A (S. 79ff) 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 $\mathbf{t}_H$ eine Hypothese und $\mathbf{t}_R$ eine Referenz, so wird für jeden Pfad $p_R \in \mathbf{t}_R$ der Pfad $p_H \in \mathbf{t}_H$ bestimmt, der die kleinste Editierdistanz $e(p_R, p_H)$ zu p_R hat.[17] Durch

$$\mathrm{wer}(\mathbf{t}_H, \mathbf{t}_R) := \frac{1}{|\mathbf{t}_R|} \sum_{p_R \in \mathbf{t}_R} \frac{\min\limits_{p_H \in \mathbf{t}_H} e(p_R, p_H)}{\max\left\{|p_R|, |\arg\max\limits_{p_H \in \mathbf{t}_H} e(p_R, p_H)|\right\}} \in [0,1]$$

wird damit ein normiertes Fehlermaß definiert. Daß Fehler im oberen Teil des Baumes implizit stärker gewichtet werden als Fehler in der Nähe der Blätter, ist nicht unbeabsichtigt, 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 $\mathrm{wer}(\mathbf{t}_H, \mathbf{t}_R)$ beim Training auf verschieden 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).

[17] Ein Baum $\mathbf{t}$ 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

$$f\colon [0,\infty) \to \mathbb{R}, \qquad f(x) := \alpha + (1-\alpha)\mathrm{e}^{-\lambda x}$$

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 $x \mapsto \alpha$, 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 Verdoppelung der Trainingsmenge der Vorsprung der Grammatik von 22,10% auf

$$1 - \frac{f(2\cdot 0{,}6247)}{\frac{1}{1{,}2210} f(0{,}6247)} = 10{,}27\%$$

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.

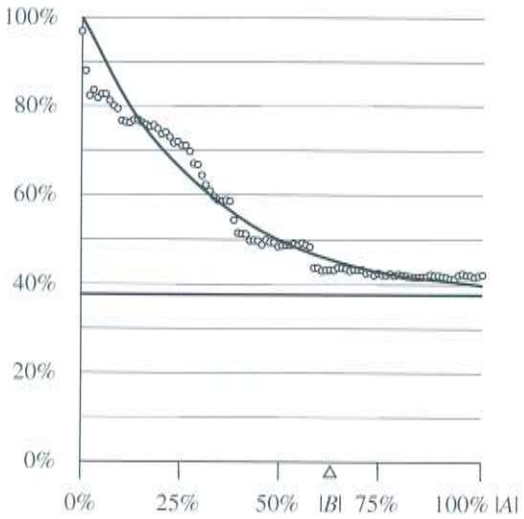

Abbildung 20 Fehlerraten bei Verwendung verschieden großer Trainingsmengen

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.

Durch die Verwendung derartig verallgemeinerter Modelle gelang es, die Methoden der statistischen Übersetzung auf die Erzeugung termartiger Interlinguarepräsentationen anzuwenden. Dieser Ansatz eröffnet eine vielversprechende Alternative zu herkömmlichen Verfahren, die zumeist auf Regelsystemen basieren, die von Experten von Hand oder semiautomatisch entwickelt wurden. Er ist ökonomischer, flexibler und langfristig vermutlich auch robuster verglichen mit grammatisch arbeitenden Systemen.

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 Übersetzungmodellen 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 erspart 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 unmodellierte strukturelle Unterschiede der Sprachpaare hervorgerufen wird.

Insgesamt zeigt die Arbeit, daß sich statistische Verfahren erfolgreich auch in Bereichen einsetzen lassen, die bisher von mehr deterministischen oder semistatistischen Ansätzen dominiert wurden. Aufbauend auf dem Fundament der reinen statistischen Übersetzung legt sie den Grundstein für Verallgemeinerungen statistischer Verfahren auf kompliziertere Datenstrukturen. Erste Verallgemeinerungen wurden exemplarisch an den unteren IBM-Modellen sowie für ein Sprachmodell und einen Decoder aufgezeigt, die für sich gemeinsam ein abgeschlossenes und anwendbares System bilden. Auf dieser Grundlage sind vielfältige Möglichkeiten zur weiteren Forschung denkbar.

Kurz- und mittelfristig wäre eine Übertragung der höheren IBM-Modelle in Betracht

zu ziehen. Der Einsatz der Fertilities im Modell 3 sollte eine bessere Ausnutzung der Symbole auf höherer Ebene bewirken, und durch die Abhängigkeit der Zuordnung von Wortklassen im Modell 4 ist eine stärkere Korrelation zu erwarten als zwischen den bloßen Positionen in Quell- und Zielsatz (vgl. die Diskussion der automatisch generierten Zuordnungen in Abschnitt 7.1, S. 59). Die Beobachtung, daß sich zusammenhängende Wortfolgen im Quellsatz häufig als Einheit einem zusammenhängenden Teilgraph des Argumentbaums zuordnen lassen, motiviert ferner die Entwicklung adaptierter Phrasenmodelle, wie sie zum Beispiel der reine statistische Übersetzer von Vogel and Ney (2000) verwendet.

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 Lizensierungen sehr leicht vermieden werden.

Abschnitt 7.5 zeigte, daß bereits einfache Modelle ihr Potential auf den 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 schätzendem 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 $\mathbf{f}$ typischerweise die Sequenz $\mathscr{A}(\mathbf{f})$ der Wortklassen $\mathscr{A}(f_1),\ldots,\mathscr{A}(f_m)$ von $f_1,\ldots,f_m$ und übersetzt diese in eine Sequenz $\mathscr{B}(\mathbf{e})$ von Zielsprach-Wortklassen: $(\mathscr{B}(e_1),\ldots,\mathscr{B}(e_l))$. Mit Hilfe einer explizit bestimmten Zuordnung $\mathbf{a}$ werden aus den Wortklassen $\mathscr{B}(e_i)$ 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 ausnutzen 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 – schlimmer – gleichmäßig auf mehrere Ebenen verteilt. Während z. B. die Zuordnung zwischen der Phrase *hier ist Stephan Vogel* (g047ak.1.3) und dem zugehörigen Term

$$\begin{aligned} \mathit{root} = (&\text{who} = (\text{given-name} = \text{name-stephan}, \text{family-name} = \text{name-vogel}),\\ &\text{communication-spec} = \text{phone}) \end{aligned}$$

unproblematisch ist (*Stephan* zu name-stephan und *Vogel* zu name-vogel), ist eine Zuordnung zwischen *hier ist* ⟨*first-name*⟩ ⟨*last-name*⟩ zu

$$\begin{aligned} \mathit{root} = (&\text{who} = (\text{given-name} = \langle \mathit{first\text{-}name} \rangle, \text{family-name} = \langle \mathit{last\text{-}name} \rangle),\\ &\text{communication-spec} = \text{phone}) \end{aligned}$$

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 $(\mathbf{f},\mathbf{e})$ der Trainingsmenge mit $\mathbf{f}=(f_1,f_2)$ und $\mathbf{e}=(e_1,e_2)$ einige der Paare $(\mathbf{f}',\mathbf{e}')$ mit $\mathbf{f}'=(f_1',f_2')$ und $\mathbf{e}'=(e_1',e_2')$ hinzufügt, für die $\mathscr{A}(f_j)=\mathscr{A}(f_j')$ und $\mathscr{B}(e_i)=\mathscr{B}(e_i')$ gilt.

Langfristig sollte man Modelle anstreben, die sich in zunehmendem Maße an den Verfahren orientieren, die bei der Sprachverarbeitung im Menschen zum Einsatz kommen. Diese Verfahren sind weder als vollständig grammatisch noch als vollständig statistisch aufzufassen, in ihnen sind vielmehr Elemente beider Theorien vereint. Nach dem Vorbild des Kindes, das eine Sprache lernt ohne je eine Grammatik zu lesen, wird man dabei nach Modellen zu suchen haben, die zunächst rein statistisch auf der Wortebene beginnen, und die mit fortschreitender Konvergenz das statistische „Grundsystem" durch erschlossenes strukturelles Wissen unterstützen. Bei einem ganz allgemeinen Vorgehen wäre auch die Interlingua a priori undefiniert und müßte anhand des Trainingsmaterials aufgebaut werden.

Zu „richtigem" Verstehen gehört wie in der Einleitung zu Kapitel 3 dargestellt neben der puren Sprachverarbeitung der sukzessive Aufbau eines mentalen Modells, in dem die sprachlich gegebene Information dauerhaft repräsentiert wird. Durch seine Fähigkeit, Information losgelöst von sprachlichen Darstellungsformen in einer abstrakten, individuell geschaffenen Welt kompakt und dauerhaft repräsentieren zu können, bleibt der Mensch für die Maschine in Sachen Sprachverstehen auf absehbare Zeit uneinholbar. Es bleibt abzuwarten, ob sich dies jemals ändern läßt.

Zum Nespole Corpus

Zur Evaluation (Kapitel 7) wurden gesammelte Daten aus dem Nespole!-Projekt verwendet. Sie bilden ein dreidimensionales Corpus von Tripeln deutscher und englischer Formulierungen sowie einer Annotation im Interchange Format, die von einem Experten stammt. Die Datenbank beinhaltet Telefongespräche (Dialoge), in denen sich ein potentieller Tourist in einer Touristeninformation über die Gegebenheiten am geplanten Reiseort informiert. Der Gesprächsparter in der Touristeninformation spricht jeweils italienisch, während die Sprache des Kunden Deutsch, Englisch, Französisch oder eine andere sein kann. Uns interessieren nur die „Dialoghälften", in denen der Kunde spricht. Die *Turns* dieser Dialoge sind von den Autoren der Datenbank bereits von Hand in *Segmente* unterteilt worden, die dem semantischen Gehalt genau eines Ausdrucks im Interchange Format entsprechen (vgl. Abschnitt 3.3). Die Originalsprache ist teils deutsch, teils englisch.

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, g022ck2, 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.

Zum Vergleich mit dem grammatisch arbeitenden System war es nötig, aus Kompatibilitätsgründen auf eine ältere Version des Interchange Formats (2. November 2001) zurückzugreifen. Für diese Version stand die Trainingsmenge B zur Verfügung, die sich aus den Dialogen zusammensetzt, die zum damaligen Entwicklungsstand vorhanden waren. Die Menge A beruht auf einer neueren Version der Datenbank, die auch Grundlage der Menge B war, so daß man sich B näherungsweise als Untermenge von A vorstellen kann – mit älteren Annotationen. Eine Beschränkung auf die Menge B als einzige Trainingsmenge erschien ungeeignet, da A um einiges größer ist und zur Testmenge C keine Annotationen in der IF-Version von B vorlagen.

Schließlich besteht die Testmenge C aus den beiden Dialogen g047ak und g051ak. Sie enthält Annotationen der neueren Version. Diese Menge wird in den Evaluationen als Menge ungesehener Daten verwendet. Die englischen Übersetzungen bzw. Interchange-Format-Annotationen dienen als Referenz.

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.

Menge	A		B		C	
Sprache	D	E	D	E	D	E
Sätze	3 885	3 885	2 427	2 427	194	194
versch. Sätze	2 135	2 113	1 395	1 395	143	141
einm. Sätze	1 970	1 951	1 283	1 290	126	123
	50,70%	50,21%	52,86%	53,15%	64,94%	63,40%
Tokens	17 434	18 248	11 236	11 729	889	955
Wörter	1 558	1 213	1 196	1 010	269	241
einm. Wörter	755	489	566	429	152	123
	48,46%	40,30%	47,34%	42,47%	56,50%	51,01%

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

Ein erheblicher Teil der Segmente sind triviale Äußerungen, z. B. *ja* oder *hallo* . Dies wird im Histogramm in Abbildung 21 deutlich. Es zeigt für Deutsch und Englisch, wie häufig Sätze jeder Länge in der Menge A auftreten. Zwei ähnliche Betrachtungen werden für das Interchange Format durchgeführt. Hier wird die Zahl der Symbole in der Wurzel sowie die Tiefe der Argumentbäume dargestellt.

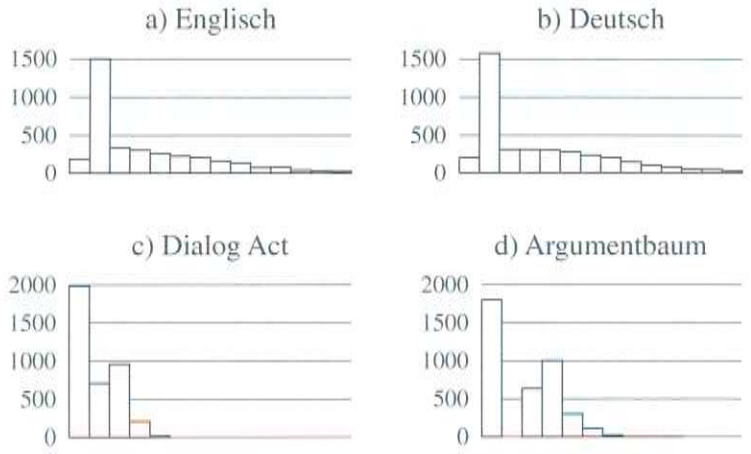

Abbildung 21 Häufigkeitshistogramme für Satzlängen und Termtiefen: a) des Englischen, b) des Deutschen, c) der Dialog Acts und d) der Argumentbäume

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.

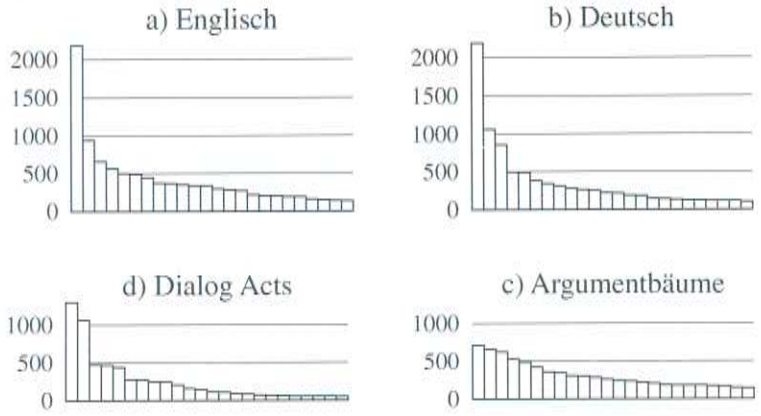

Abbildung 22 Häufigkeitshistogramme für Wort- bzw. Symbolvorkommen: a) im Englischen, b) im Deutschen, c) in den Dialog Acts und d) in den Argumentbäumen

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]

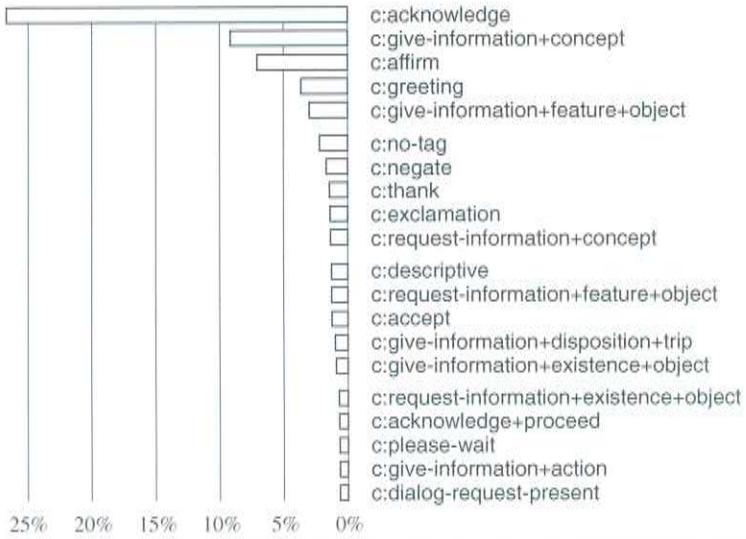

Abbildung 23 Liste der häufigsten Dialog Acts zusammen mit ihren relativen Häufigkeiten

[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

In diesem Anhang werden die Übersetzungen präsentiert, die für die Humanevaluation (Kapitel 7) von den verschiedenen Systemen generiert wurden. Aus Platzgründen sind hier nur die nichttrivialen Segmente aufgeführt. Ein Segment wurde dabei als nichttrivial betrachtet, wenn es ohne Interpunktionszeichen im Original mindestens zwei Worte umfaßt und nicht in allen Systemen zum gleichen Resultat führt. Überdies wurden Duplikate gelöscht. Für die vollständigen Datensätze sei auf die CD (vgl. S. 105) verwiesen.

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

1. die Übersetzung eines menschlichen Experten
2. die Übersetzung des statistischen Systems von Stephan Vogel (vgl. Vogel and Ney, 2000). Die *kursiv* gesetzten Wörter waren dem System unbekannt und wurden ohne Bearbeitung in die Übersetzung übernommen.
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 statistischen Systeme wurden auf der Trainingsmenge *B* (vgl. S. 79) trainiert, und zur Generierung von Englisch aus IF in 3., 4. und 5. wurde das gleiche System verwendet.

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

Segment-ID	**Original** **Rein statistische Übersetzetzung** **JTrans, trainiertes Modell**	**Humanübersetzung** **Rein grammatische Übersetzung** **JTrans, projiziertes Modell**
g047ak.1.3	*hier ist Stephan Vogel.* here is *Stephan Vogel.* It is.	this is Stephan Vogel. Stephan's bird speaking. This one is something.
g047ak.2.1	*ich würde gerne Urlaub machen.* i would like to take a vacation. I want to travel for vacation.	I would like to go on vacation. I want to travel for vacation. I want to travel for vacation.
g047ak.3.2	*und zwar Skiurlaub.* more precisely Skiurlaub. Specifically, That.	specifically skiing vacation. Specifically, That. And.
g047ak.4.1	*im Dezember,* in December to The person December.	in December, December. I am favorably disposed towards traveling December.
g047ak.5.2	*ich hab' noch keinen Ort.* I also isn't place.	I don't have a location yet. I would have place.

	But I would not have another place.	Not.
g047ak.6.2	*Skifahren ist gut.* skiing is good. It is good.	skiing is good. Skiing would be good. It is good.
g047ak.7.3	*eine Sache dabei,* daycare decide among to I am travelling.	one thing with that, . Because of that, is there something?
g047ak.7.4	*ich habe zwei Kinder.* i asked two children. It would cost something.	I have two children. I have 2 children. A child is favorably disposed towards doing something.
g047ak.7.6	*wo Kinder auch ganz gut fahren können,* where children homepage relatively good which you said It is somewhat good skiing.	where kids can go pretty well too, Where is it possible to see children's good one also? Can child also go skiing?
g047ak.7.7	*oder Anfänger.* or *Anfänger.* That.	or beginners. Or. That.
g047ak.8.2	*gibt es Unterkunft mit Küche,* is there accommodation include kitchen to There would be something for accommodation with ones.	is there accommodation with kitchen, Kitchen with accommodation? There would be accommodation with them.
g047ak.8.3	*daß man auch selbst etwas kochen kann?* so you homepage even something *kochen* can get Then inhabitant?	so that you can cook something yourself? Then it is possible to cook something even also. Also, something can?
g047ak.11.1	*können Sie mir noch sagen,* can you tell nothing tell, Please tell again.	could you also tell me, You can tell something to me. Please tell me.
g047ak.11.2	*ob das Apartments mit zwei Schlafräumen sind?* whether the *Apartments* with two *Schlafräumen* are? Would it cost something for a room with the discount for the children?	if that is apartments with two bedrooms? The apartments. With children?
g047ak.12.1	*mit zwei Schlafräumen,* with two *Schlafräumen* to That.	with two bedrooms, Two. With child.
g047ak.12.2	*für Eltern, für Kinder.* for parents, for children. It would cost something for a room.	for parents, for kids. ??? It would cost something for the child.
g047ak.14.1	*sehr schön.* very nice. Very nice one.	very nice. Okay. It is very nice.
g047ak.15.1	*hätte ich gerne,* I'd i like to I want something.	I would like to have, I want to do something. I want to travel.
g047ak.16.2	*vielen Dank.* thank you very much. Thank you.	thank you very much. Thank you. Thank you.
g047ak.19.2	*und da ist ein Skipaß verfügbar für die Woche?* and there is a *Skipaß verfügbar* for the week? And is a week the one?	and is there a ski pass available for the week? Because it is something. And is there a week?
g047ak.20.2	*gibt es da Ermäßigung für Kinder?* are there discount for children? Would it cost something for the one for children with discount there?	is there a discount for children? Would there be discount for children? Would there be something for children?
g047ak.21.2	*und gibt es die Möglichkeit im Hotel oder in einer Skischule Skier auszuleihen?* 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?	and is there a possibility to rent skis in the hotel or in a ski academy? And the option at the hotel? Do I rent skis? And would there be hotels?
g047ak.22.3	*vielen Dank.* thank you very much. Thank you.	thank you. Thank you. Thank you.
g051ak.1.2	*guten Tag.* goody day tour	good day. Good day!

	Good day!	Good day!
g051ak.1.3	*Sebastian Schlüger hier.* *Sebastian Schlüger* list here I see.	Sebastian Schlueger here. Here. It is something.
g051ak.2.1	*bin ich da mit der Touristenauskunft verbunden?* am I right include the *Touristenauskunft verbunden*? The one there on the foot?	am I connected with the tourist information? I. The one. I am interested in traveling with them.
g051ak.3.1	*ah, sehr schön.* ah, very nice. It is very nice.	ah, very nice. Okay. It is very nice.
g051ak.3.3	*ich interessier' mich für einen Winterurlaub in der Nähe von* i'm interested in a winter vacation in the nearby of I am interested in traveling for a winter vacation nearby.	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.
g051ak.3.4	*oder in Valle-di-Fiemme* or in Valle-di-Fiemme Or it is at Val di Fiemme in Val di Fiemme.	or in Val-di-Fiemme Or at Val di fiemme official area. Or is that it?
g051ak.3.5	*und suche jetzt 'ne Unterkunft.* and looking now suite accommodation. I am interested in accommodation.	and I am looking for an accommodation. And I am interested in an accommodation. And I am interested in accommodation now.
g051ak.4.1	*und zwar wollte ich am elften Dezember anreisen.* more precisely wanted I the eleventh of car. I am favorably disposed towards traveling December the 11th.	specifically, 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.
g051ak.5.1	*ist das möglich?* is that possible? Is it possible to?	is that possible? Is it possible to? Something?
g051ak.6.1	*ah ja,* ah yes, Okay.	I see, Okay. I see something.
g051ak.6.2	*sehr gut.* very good. It is very good.	very good. Okay. It is something.
g051ak.7.1	*den Lauf hab' ich schon* catch *Lauf* asked i important I see information.	I already have the run I would have the foot already. Could I have information?
g051ak.8.3	*also ich interessier' mich hauptsächlich für Alpin-Skilaufen.* well i'm interested in mostly interested *Alpin-Skilaufen*. The person for me is interested in traveling for the one.	well I am mainly interested in alpine skiing. I am interested in going skiing. I am interested in traveling.
g051ak.8.4	*würde aber nichts ausmachen,* would but nothing *ausmachen* to However, I see something.	but wouldn't matter, However, nothing will be available. However, it is not.
g051ak.8.5	*auch wenn ich mal hin und wieder Langlaufen könnte.* even if i extend goes and it's cross-country guided tours If I.	also if I could do cross country skiing every once in a while. When something again. And if I can do something something.
g051ak.10.1	*wie sind die Skigebiete über Busse miteinander verbunden?* how are the areas about *Busse miteinander verbunden*? Tell me about.	how are the skiing areas connected by bus? How is it the ski areas how? Information attractions.
g051ak.11.2	*wie sieht das aus,* what about the slope, Tell me about.	what about this, How is it the one how? What is it?
g051ak.11.3	*kann man das Skigebiet als familienfreundlich bezeichnen?* can you, ski than *familienfreundlich bezeichnen*? And is it possible to do something?	can you call the skiing area suited for families? When it is possible to do something. You can?
g051ak.11.5	*ich werde* i will That's a promise!	I will I. I want something.
g051ak.11.7	*mit meiner Frau und zwei Kindern, drei und sieben, anreisen.* take my wife and two children, three and seven, car. And daughter, on foot.	come with my wife and two children, three and seven. 3, and 7 outgoings with my adult, and with 2 children. .

g051ak.12.1	*also der ältere hat schon mal einen Skikurs mitgemacht,*	well the older one has participated in a skiing course once before,
	so the *ältere* Until already extend a course *mitgemacht* to	The older one.
	I am favorably disposed towards the one already.	I want the one.
g051ak.12.2	*aber die Jüngste,*	but the youngest one,
	but the *Jüngste* to	However, the most young one.
	The one.	However, it is something.
g051ak.12.3	*die müßte vielleicht erst,*	she would maybe first have to,
	they work maybe first to	It is possible that I must reserve the one only.
	It is possible that.	It is possible that it is the one.
g051ak.12.4	*und das ist vielleicht noch ein bißchen jung,*	and that is maybe a little young still,
	and that is might have a little bit young to	Okay.
	And it is something.	And is there?
g051ak.12.5	*die müßte in den Skikindergarten.*	she would need to go in the skiing kindergarten.
	they work in the *Skikindergarten.*	I require the one, the one.
	Correct, the hotel at the hotel.	It is at the hotel.
g051ak.13.3	*das wär' natürlich sehr schön.*	that would of course be very nice.
	that would be course very nice.	It is very nice.
	It is very nice September.	It is very good.
g051ak.14.2	*wie sieht das aus mit der Verpflegung?*	what about the food?
	what about the slope with the Verpflegung?	How the food how.
	Tell me about that one at the one.	Accommodation.
g051ak.14.4	*besteht da die Möglichkeit, die Kinder dann auch*	is there the possibility to then also the kids
	besteht that the possibility to the children Am-Fasanengarten also	The option then also.
	Children options there?	Also, would there be attractions?
g051ak.14.5	*daß die mittags verpflegt werden?*	that they get fed at noon?
	noticed the *mittags verpflegt* will get	Then I will feed in the mid-day.
	Then them?	Will it be the one?
g051ak.15.3	*meiner Frau*	my wife
	mother wife	My wife.
	Female.	I am travelling.
g051ak.15.5	*der ist es auch ganz lieb,*	she would also somewhat like it,
	spread is it also poorly friendly to	Also, it is the one.
	Friendly one is somewhat nearby.	Also, is there the one?
g051ak.15.6	*wenn sie dann mal im Hotel verwöhnt wird,*	if she would get pampered in the hotel some time.
	if they then extend the hotel *verwöhnt* will.	When you will reserve a room at the hotel.
	If it is possible to do something something.	If I am favorably disposed towards accommodation at a hotel.
g051ak.16.2	*so zwei bis*	about two to
	so 2 driving	Two.
	That.	That.
g051ak.18.2	*also mir persönlich ist das schon wichtig.*	well for me personally that is pretty important.
	so Please *persönlich* is that be important.	Me.
	It was important.	It is good.
g051ak.18.3	*wir werden mit dem Auto ankommen.*	we will arrive by car.
	we will come by car.	We will arrive.
	We are arriving by car.	We are arriving by car.
g051ak.18.4	*und da das ja 'n Winterurlaub ist,*	and because this is a winter vacation,
	and there settle ues picture vacation is,	Because it is a winter vacation.
	And image is there.	And it is there.
g051ak.18.5	*besteht die Möglichkeit, das Auto in einer Garage unterzubringen,*	is there the possibility to keep the car in a garage
	besteht the possibility to the car in a *Garage unterzubringen* to	The option with the car at a garage.
	Option at the one is at the bus stop.	Is it possible to go?
g051ak.19.2	*das ist mir schon klar.*	i understand that.
	that is Please important course.	Is that right?
	It is already.	It is good.
g051ak.20.1	*nur, steht das Auto dann in einer Garage*	but is the car in a garage then
	just to *steht* the car Am-Fasanengarten in a *Garage*	The car at a garage only.

	The person at the one only is favorably disposed towards traveling at bus stop.	I am favorably disposed towards traveling at car only.
g051ak.23.2	*das is okay,* that is okay, Here it is.	that is alright, It is something. Okay. It is something.
g051ak.23.3	*das ist sehr gut.* that is very good. It is very good.	that is very good. That sounds good. Okay. It is very good.
g051ak.24.2	*ich schaue grade.* i *schaue grade.* I am favorably disposed towards reserving a trip.	I am looking right now. I see now. I am favorably disposed towards traveling.
g051ak.24.3	*also 'n 'n drei Sterne sollt' es schon sein.* so picture picture three stars should it important fifteenth of Three star people at image shows something already.	well it should at least be three stars. I must rent. The three star one would be good.
g051ak.24.4	*zwei Sterne wären in Ordnung,* two stars be okay, Three star is and at day.	two stars would be fine, Is it two star? That sounds good. I want to stay at the three star one.
g051ak.24.5	*aber das* but that That.	but that However, the one. But.
g051ak.24.6	*das wär' dann entweder das.* that would be then entweder that. Them are good.	that would then either be that one. Or. If so, it is something.
g051ak.24.8	*das da,* that one, Is that it there?	that one there, Okay. Is there the one there?
g051ak.24.9	*oder welches war das davor?* or which was that *davor*? Or would the accommodation be the one?	or which one was the one before? Or is it earlier? Or is that it?
g051ak.25.2	*das Rio-Bianco würde mich interessieren.* that Rio-Bianco would interest actually. I am interested in something.	the Rio-Bianco would interest me. I am interested in Rio bianco hotel. I am interested in accommodation.
g051ak.27.2	*das klingt interessant.* that sounds interesting. It is too good.	that sounds interesting. It is interesting. It is very interesting.
g051ak.28.1	*ich sagte,* i *sagte* to I.	I said, I. I am favorably disposed towards traveling.
g051ak.28.4	*das würde mich interessieren.* that would interest actually. I am interested in something.	that would interest me. I am interested in something. I am interested in traveling.
g051ak.29.1	*das klingt* that sounds That sounds good.	that sounds Is it the one? It is something.
g051ak.30.1	*das klingt auch interessant.* that sounds homepage interesting. It is too good.	that also sounds interesting. It is interesting. It is interesting.
g051ak.30.2	*das wäre vielleicht nicht schlecht.* that might isn't bad again It is possible that.	that would maybe not be bad. It is possible that it is not the one. It is possible that the bad one.
g051ak.32.2	*das klingt interessant.* that sounds interesting. It is too good.	that sounds interesting. It is interesting. It is very interesting.
g051ak.33.2	*das ist natürlich* that is sure The one is something.	that is obviously It is natural. The one is something.
g051ak.33.4	*meine Frau hat mir hier* my wife Until me here Your daughter here.	my wife has me here My wife would have transportation to me here. It is possible to go.
g051ak.35.2	*was mich jetzt erst mal noch interessieren würde,*	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.
g051ak.35.3	*meine Frau, hab' ich gesehen, hat mir aufgeschrieben, ihre bevorzugten Hotels.* my wife, asked i noticed to Until me *aufgeschrieben* to your *bevorzugten* hotels. Apparently, hotel's daughter hotel.	my wife, I saw, wrote down for me her preferred hotels. My wife prefered to reserve your one to me. My child can do something.
g051ak.35.4	*um dazu entscheiden zu können,* 2pm *dazu entscheiden* to Could to So that you can take it.	in order to be able to decide in that respect, So that something. Is it possible to go?
g051ak.35.5	*die Skigebiete,* the ski school It is something.	the skiing areas, The ski areas. It is at the ski area.
g051ak.35.6	*haben Sie detaillierte Informationen,* do you *detaillierte* information available You would have the information.	do you have detailed information, Would you have detailed information? Could there be the information?
g051ak.35.7	*wo welche Liftanlagen sind,* where which forty-one are, Which ski slope?	where which ski lifts are located, Where is the facility ski lifts? Which one is there?
g051ak.35.9	*bei den* to the That.	at the The one. That.
g051ak.37.1	*das wäre nett.* that would be nice. It is something.	that would be nice. That sounds good. It is nice.
g051ak.41.1	*machen wir das so.* do it that way. That sounds good.	we're doing it that way then. Yes, i do. Because of that, we are favorably disposed towards doing something.
g051ak.42.1	*wie sieht das aus,* what about the slope, Tell me about.	what about this, How is it the one how? What is it?
g051ak.42.3	*damit ich ein Hotel noch bekomme?* *damit* find a hotel nothing *bekomme*? Am I favorably disposed towards a hotel?	so that I will still get a hotel? So that I am favorably disposed towards a hotel. Am I favorably disposed towards a hotel?
g051ak.44.2	*was ich noch fragen wollte,* something else ask, there I am doing something else called Predazzo.	what I still wanted to ask, What I want to ask for. Am I doing something?
g051ak.44.3	*hin und wieder mach' ich auch gerne Eislauf.* goes and it's make i also like *Eislauf.* And the person also wants to travel March.	off and on I also like to do ice skating. Also, ice skating again. And I want to do something.
g051ak.44.4	*gibt es genügend Eislaufgebiete* are there *genügend Eislaufgebiete* Also, Is there?	are there enough ice skating areas Is there? Is there something there?
g051ak.44.5	*Eislaufringe da* *Eislaufringe* there That?	ice rinks there ? Is there?
g051ak.45.2	*können Sie mir vielleicht dann auch gleich noch detaillierte Hotelprospekte zuschicken.* can you maybe then also coming have *detaillierte Hotelprospekte* send. It is possible that you also can send brochures.	could you probably also send me detailed hotel brochures then. Can I, you see same detailed one also? It is possible to send brochures.
g051ak.45.3	*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, My wife would have transportation to me here. Please take at your here.
g051ak.45.4	*daß ich* that I Undefinable-descriptive-tag	that I Then I. ???
g051ak.45.5	*daß sie gerne aus Molina was hätte,* noticed they like Excelsior Molina what I'd to Then the how many of them you want.	that she would like to have something from Molina, Then is from Molina. Do you want something?

g051ak.45.6	*vom Hotel Latemar.* the hotel *Latemar*. From the hotel from the hotel.	from the hotel Latemar. From Latemar hotel. Accommodation from a hotel.
g051ak.46.1	*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.	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.
g051ak.47.2	*das wäre sehr schön.* that would be very nice. It is very good.	that would be very nice. It is very nice. It is very nice.
g051ak.47.3	*und vielleicht auch* and maybe also And also, it is possible that.	and probably also And same here. And it is possible that.
g051ak.47.4	*ja gut.* okay. Okay.	okay. Okay. Okay.
g051ak.47.5	*dann gleich noch 'ne Frage speziell zum Alpin* then coming nothing suite question particular Hotel *Alpin* At downhill skiing to question specifically then.	then also a question on the alpine Specifically, same question the Alpine one. Another question.
g051ak.47.6	*wo wir* where feed Is it possible?	where we Where we. Is it possible?
g051ak.47.7	*wo ich die Kinder jetzt untergebracht habe,* where i the children now *untergebracht* asked to Can you see anything at the one now?	now that I have put up the kids, I have the children now. Can you see anything now?
g051ak.48.2	*für mich persönlich einen Tiefschneekurs.* for me *persönlich* a *Tiefschneekurs*. And I am favorably disposed towards traveling for a child.	for me personally a deep snow course. One for me. I am interested in accommodation.
g051ak.49.3	*und wie sieht das aus mit Privat-Skilehrern?* and what about the slope include *Privat-Skilehrern*? And who would please take on that bus.	and what about private ski instructors? And with what payment method do I pay? Is it the one? And accommodation.
g051ak.49.4	*könnten Sie mir da auch gleich ein Prospekt zusenden mit Preisen?* could you send me there also coming a brochure *zusenden* with *Preisen*? Information concerning you there something containing a there.	could you also send me a brochure then with prices? – I F formatting error – Also, can you send information?
g051ak.52.2	*dann glaub' ich* then think i That.	then I think It is probable that. I am favorably disposed towards traveling.
g051ak.52.3	*hab' ich so* asked i so I have it.	I have so Would I have something? I would have information.
g051ak.52.4	*haben wir soweit alles abgeklärt* do feed *soweit* everything *abgeklärt* We plan on everything.	we have settled everything so far Do we have everything? We must take.
g051ak.52.6	*wenn ich mich jetzt entschließen sollte,* if i contact now *entschließen* date should If date is something.	if I should decide now, When I am favorably disposed towards traveling. If the person now sees something.
g051ak.52.7	*keine Halbpension zu nehmen,* downhill board to take, Not.	not to take half-board, I accept half board. I want half board.
g051ak.52.8	*sondern das Hotel nur mit Garni,* start the hotel only with *Garni* to I can start containing the hotel only.	but just the hotel with breakfast, Breakfast only. Hotel only.
g051ak.52.10	*wenn man sich das individuell zusammen stellen möchte.* if you somehow that individuell together questions go on And the good place is something.	if you want to put it together individually. When it is something, right? If you want something.
g051ak.52.11	*also nicht in Halbpension gehen,* so not in board go to	so not to take half-board, On foot at half board.

	Not on foot.	I want to do something at half board.
g051ak.52.12	*sondern dann je nach dem auf der Hütte essen.*	but instead then depending on that just eat on a hut here and there.
	start then *je* after the on the *Hütte* eat.	???
	I will do it.	I am favorably disposed towards traveling to the food.
g051ak.54.1	*wieviel wäre das in Euro?*	how much would that be in Euro?
	how much would that be in *Euro*?	???
	If is that it?	Is that it?
g051ak.57.3	*dann bedank' ich mich bei Ihnen.*	then I thank you.
	then thank you do you.	Thank you!
	If so, you.	You are interested in traveling.

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 Virtuellen 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:

⟨command⟩ → ⟨expr⟩ | ⟨def⟩ | ⟨command⟩ **';'** ⟨command⟩.
⟨def⟩ → ⟨id⟩ **'='** ⟨expr⟩.
⟨expr⟩ → ⟨id⟩ | ⟨literal⟩ | ⟨id⟩⟨args⟩ | ⟨expr⟩ **':'** ⟨id⟩⟨args⟩ | ⟨expr⟩ **'+'** ⟨expr⟩.
⟨args⟩ → **'()'** | **'('** ⟨exlist⟩ **')'**
⟨exlist⟩ → ⟨expr⟩ | ⟨expr⟩ **','** ⟨exlist⟩.

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 boolschen Werte, die Zeichenketten, die von den Konstruktoren der Klassen `java.lang.Integer` und `java.lang.Double` als `int` bzw. `double` erkannten werden, sowie Stringkonstanten. 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 Zeilenende gehen oder sich von **'/*'** bis zu **'*/'** erstrecken.

Ebenfalls ignoriert wird das Token **'new'**. Zur Unterscheidung zwischen Konstruktoraufrufen und Funktionsaufrufen wird statt eines voranstehenden **'new'** die Form des Bezeichners herangezogen. So es sich dabei entsprechend der in Java üblichen Konventionen um einen Klassennamen handelt, d. h. ein Großbuchstabe gefolgt von Kleinbuchstaben, Ziffern und/oder Großbuchstaben und möglicherweise vorangestellten Paketbezeichnern, die dann durch Punkte voneinander zu trennen sind, wird der Aufruf als Konstruktur interpretiert, in allen anderen Fällen als Funktion. So werden also z. B. `"Corpus"` und `"nespole.TreeIbm1TM"` als Klassennamen und `"exit"` als Funktionsnamen erkannt. Diese Regelung mag zunächst etwas gewöhnungsbedürftig erscheinen, erweist sich aber in der Praxis als wenig störend oder gar bequem.

Zur *Auswertung*: Nachdem über einem eingegangenen 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 Methodenaufruf werden ausgeführt, sobald alle nötigen Argumente ausgewertet wurden. Die Addition ⟨expr⟩ **'+'** ⟨expr⟩ wird behandelt wie ⟨expr⟩ **':add('** ⟨expr⟩ **')'**. Für den Aufruf von Methoden stehen alle öffentlichen (public), nicht-statischen Methoden zur Verfügung, die für das Objekt, auf dem sie ausgeführt werden sollen, vorhanden sind. Falls die Parameter und der Bezeichner bei einem Methodenaufruf zu mehreren Signaturen passt, wird eine Disambiguierung 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), Fehlerausgaben (err) sowie Protokollausgaben (log).

Beim Programmstart werden out und err durch `java.lang.System.out` und `java.lang.System.err` initialisiert, und log führt ins Nichts bzw. in die Datei, die beim Programmstart 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

Mit den Schreib- und Leseroutinen des Systems lassen sich sowohl Text- als auch Binärdateien verarbeiten. Die Klassen *LineWriter* und *LineReader*, die im Paket `io` definiert werden, dienen zur zeilenweisen Verarbeitung von Textdateien. Analog dazu sind *IntWriter* und *IntReader* für die Verarbeitung von Binärdateien zuständig. An die Stelle der Zeilen treten hier Integer-Werte, d. h. es werden jeweils vier Bytes gemeinsam geschrieben oder gelesen. Neben bloßen Integer-Werten können auch Werte der Typen double, boolean, long und String ohne zustätzlichen Aufwand behandelt werden. Die nötige Umwandlung in Folgen von Integern wird von den Klassen geleistet. Die Binärdateien, die von *IntWriter* und *IntReader* behandelt werden, werden standardmäßig in komprimierter Form auf dem Datenträger abgelegt.

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.

Für die Speicherung von Konzepten existiert ein einheitlicher Mechanismus. Die Speicherung wird über die Methode *saveTo*() durch Angabe eines Dateinamens angestoßen. Sie erzeugt einen Deskriptor und läßt ihn durch die abstrakte Methode *updateDescriptor*() mit Definitionen bestücken. Wenn Konzepte ineinander enthalten sind (z. B. enthält ein *LanguageModel* ja ein *Lexicon*, vgl. Abb. 14 auf S. 54 oder Abb. 28 auf S. 97), enthält der Deskriptor Referenzen zu den Deskriptor-Dateien, die die enthaltenen Konzepte beschreiben. Die dazu nötigen Dateinamen können über die Methode *getFilename*() in Erfahrung gebracht werden. Wenn *getFilename*() auf einem Konzept ausgeführt wird, zu der noch keine Deskriptor-Datei vorliegt, oder wenn das Konzept seit der letzten Speicherung verändert wurde, stößt diese Methode ihrerseits eine Speicherung an, nötigenfalls wird automatisch ein Dateiname generiert. Im *Descriptor* aus Abb. 24 war das *Lexicon* zum Beispiel bereits vor dem Sprachmodell unter `eng.lex` gespeichert worden, während für die Wahrscheinlichkeitstabelle `14652.probs` kein Dateiname bekannt war und automatisch ein Dateiname generiert wurde.

Datenstrukturen

Die Counts von n-Grammen und den diversen Wahrscheinlichkeitsverteilungen werden in Instanzen der Klasse *TupleMap* abgelegt. Genauer ist *TupleMap* die gemeinsame Oberklasse von *IntTupleMap* und *DoubleTupleMap*, die eine Abbildung von int-Tupeln auf Integer- bzw. Double-Werte implementieren.

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 $O((\log k)^n)$ 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

Die Klassen des Systems JTrans sind in sechs Paketen organisiert, wovon zu vieren in den Abbildungen 25 bis 28 eine Übersicht gegeben wird. Die Diagramme erheben keinen Anspruch auf Vollständigkeit und sollen lediglich zur Orientierung dienen. Nebensächliche Klassen und Methoden sind nicht aufgenommen und auf die Angabe von Argumenten, Stelligkeiten und Rückgabewerten von Methoden wurde aus Platzgründen verzichtet. Einzelheiten sind in elektronischer Form verfügbar (vgl. S. 105).

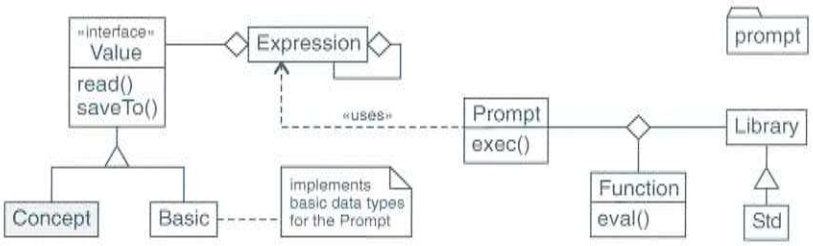

Abbildung 25 Klassen des Pakets `prompt`

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-

plementierung eine Klasse als lesbar oder schreibbar deklariert wird. Ferner werden die Klassen zum Lesen und Schreiben von Binärdateien und zeilenorientierten Dateien sowie die Klasse *Descriptor* (s. o.) bereitgestellt. Nicht abgebildet ist die Klasse *OutputManager*, die die drei permanenten Ausgabeströme (s. o.) verwaltet.

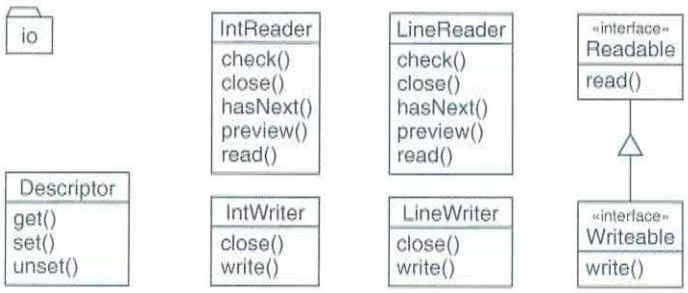

Abbildung 26 Klassen des Pakets `io`

Das Paket `util` beinhaltet eine weitgehend unzusammenhängende Sammlung häufig gebrauchter Klassen und Datenstrukturen. Hierzu gehören die *TupleMap* (s. o.) sowie verschiedene Implementierungen von Iteratoren, Mengen, Multimengen, Warteschlagen usw.

Die Pakete `prompt`, `io` und `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 `base` und `concepts`, die Klassen zur Behandlung von Termsprachen im Paket `nespole`.

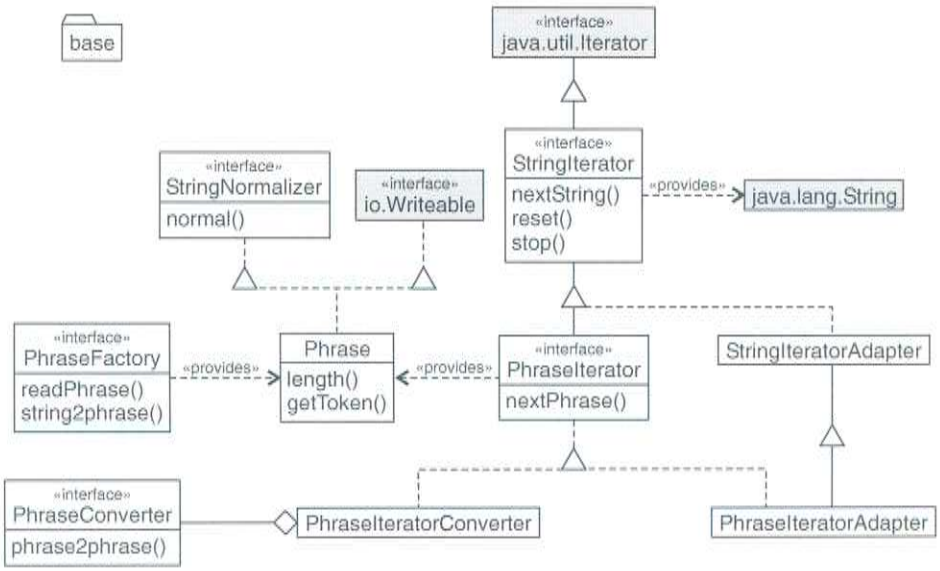

Abbildung 27 Klassen des Pakets `base`

Im Paket `base` wird die Klasse *Phrase* zur Codierung von Sätzen definiert. Die Klasse ist eingebunden in eine Reihe von Werkzeugen, mit denen Phrasen und Mengen von Phrasen verarbeitet werden können. Zentrale Bedeutung haben dabei die Iteratoren *StringIterator* (zur Aufzählung von Zeichenketten) und *PhraseIterator* (zur Aufzählung von Phrasen). Dimensionen eines *Corpus* werden in Form solcher Iteratoren bereitgestellt. *StringIteratorAdapter* und *PhraseIteratorAdapter* implementieren *StringIterator* und *PhraseIterator* trivial.

Die Schnittstelle *PhraseFactory* zur Erzeugung von *Phrase*-Objekten wird z. B. von *Lexicon* und *PhraseConverter* zur Umwandlung von Phrasen z. B. von *TranslationModel* implementiert.

Die grundlegenden Klassen, deren Objekte von der Benutzerstelle aus zugreifbar sind, sind im Paket `concepts` zusammengefaßt. Die Klassen dieses Pakets wurden bereits in Kapitel 6 (S. 53) vorgestellt. Abbildung 28 zeigt eine ausführlichere Version von Abbildung 14. Eine Übersicht über das Paket `nespole` findet man auf Seite 57.

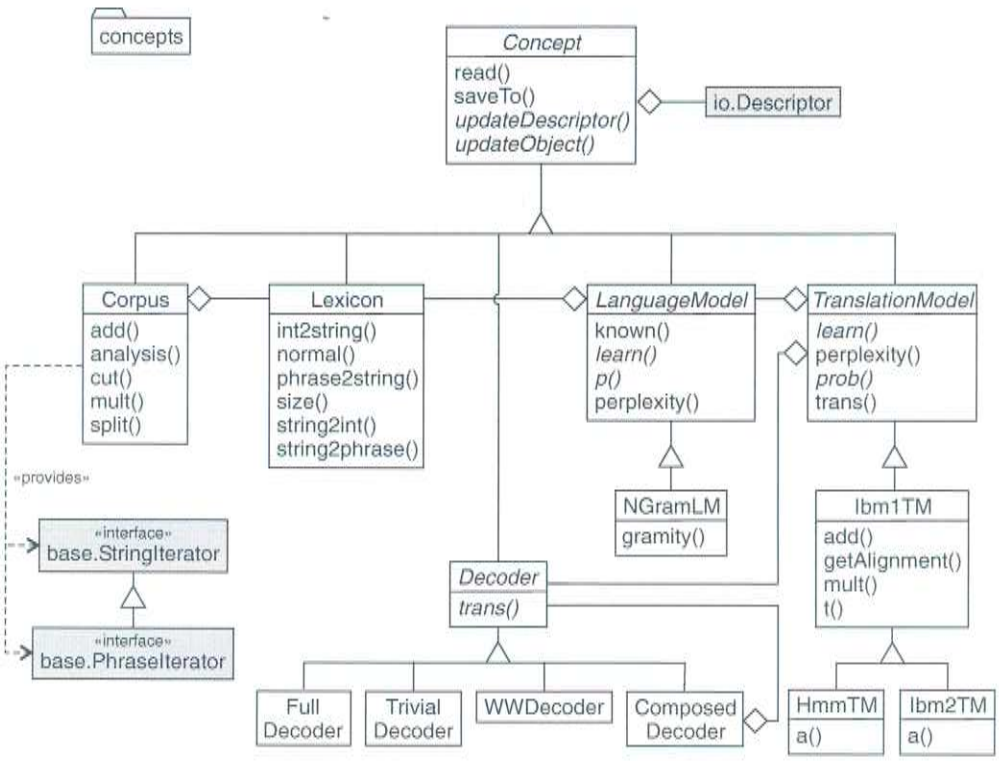

Abbildung 28 Klassen des Pakets `concepts`

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:

$$
\begin{aligned}
\langle \text{top} \rangle &\rightarrow \langle \text{root} \rangle \mid \langle \text{root} \rangle \langle \text{tree} \rangle . \\
\langle \text{root} \rangle &\rightarrow \langle \text{id} \rangle \texttt{':'} \langle \text{concepts} \rangle . \\
\langle \text{concepts} \rangle &\rightarrow \langle \text{id} \rangle \mid \langle \text{concepts} \rangle \texttt{'+'} \langle \text{id} \rangle . \\
\langle \text{tree} \rangle &\rightarrow \texttt{'('} \langle \text{args} \rangle \texttt{')'} . \\
\langle \text{args} \rangle &\rightarrow \langle \text{id} \rangle \texttt{'='} \langle \text{val} \rangle \mid \langle \text{id} \rangle \texttt{'='} \langle \text{val} \rangle \texttt{','} \langle \text{args} \rangle . \\
\langle \text{val} \rangle &\rightarrow \langle \text{id} \rangle \mid \langle \text{tree} \rangle \mid \langle \text{set} \rangle . \\
\langle \text{set} \rangle &\rightarrow \texttt{'['} \langle \text{vals} \rangle \texttt{']'} \mid \texttt{'('} \langle \text{coord} \rangle \texttt{')'} . \\
\langle \text{vals} \rangle &\rightarrow \langle \text{val} \rangle \mid \langle \text{val} \rangle \texttt{','} \langle \text{vals} \rangle . \\
\langle \text{coord} \rangle &\rightarrow \langle \text{val} \rangle \mid \langle \text{val} \rangle \texttt{'\&'} \langle \text{vals} \rangle .
\end{aligned}
$$

Gültige Bezeichner ⟨id⟩ sind Zeichenketten, die keine Leerzeichen und keines der Zeichen `':'`, `'+'`, `'&'`, `','`, `'='`, `'('`, `')'`, `'['`, `']'` enthalten. Groß- und Kleinschreibung wird nicht unterschieden. Leerzeichen zwischen Terminalsymbolen werden überlesen. Die *Coordination* ⟨coord⟩ ist nicht mehr Teil des Interchange Formats, wird aber aus historischen Gründen noch weiter berücksichtigt. Semantisch wird (⟨val⟩&...) wie [⟨val⟩,...] behandelt.

Während der Syntaxanalyse werden Wurzel und Argumentbaum aufgebaut, zur Darstellung der Knoten wird ihr Index innerhalb des *IFLexicon* verwendet. Aus jeder Zeichenkette, die der obigen Grammatik genügt, wird eine zugehörige *IFPhrase* korrekt aufgebaut. Semantische Korrektheit ist an anderer Stelle zu prüfen. Wenn die Analyse einer Zeichenkette zum Aufbau einer *IFPhrase* an einem syntaktischen Fehler scheiterte, wird eine Ausnahme mit einer Fehlermeldung ausgelöst, die den Fehler beschreibt.

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

Die Datei `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

$$
\begin{aligned}
\langle \text{def} \rangle &\rightarrow \texttt{'('} \langle \text{id} \rangle \langle \text{conts} \rangle \langle \text{args} \rangle \texttt{')'} . \\
\langle \text{conts} \rangle &\rightarrow \texttt{'nil'} \mid \texttt{'('} \langle \text{cont} \rangle^{+} \texttt{')'} . \\
\langle \text{cont} \rangle &\rightarrow \langle \text{id} \rangle \mid \texttt{'('} \langle \text{id} \rangle^{+} \texttt{')'} . \\
\langle \text{args} \rangle &\rightarrow \texttt{'('} \langle \text{id} \rangle^{+} \texttt{')'} .
\end{aligned}
$$

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 Attributunterargumente), `':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

$$\begin{aligned}
\langle \text{arg} \rangle &\rightarrow \text{'('} \langle \text{id} \rangle \langle \text{def} \rangle^{+} \text{')'}. \\
\langle \text{def} \rangle &\rightarrow \text{'('} \langle \text{list} \rangle^{+} \text{')'}. \\
\langle \text{list} \rangle &\rightarrow \langle \text{slot} \rangle \text{'('} \langle \text{id} \rangle^{+} \text{')'}. \\
\langle \text{slot} \rangle &\rightarrow \text{':ISA'} \mid \text{':VALUES'} \mid \text{':ATTRIBUTES'} \\
&\quad \mid \text{':RELATIONS'} \mid \text{':COMMENTS'} \mid \text{':GOTO'}.
\end{aligned}$$

erwartet. Die Elemente einer `':COMMENTS'` Liste können Stringliterale sein, in denen zwischen zwei `'"'` Symbolen beliebiges Material eingschlossen werden kann. In der Implementierung des *IFLexicon* ist die Reihenfolge der Listen egal, und ein `':GOTO'` schließt zusätzliche Listen nicht aus.

Schließlich definiert `nespole-values.db.lsp` die atomaren Werte. Werte können andere Werte beinhalten, man spricht dann von *Werteklassen*. Diese Klassen sind nur innerhalb der Spezifikation von Bedeutung, sie werden bei der Lizensierung von atomaren Werten durch Argumente eingesetzt (Liste `':VALUES'`, s. o.). Bezeichner von Werten, die andere Werte beinhalten, sind durch umgebende Sterne `'*'` gekennzeichnet.

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 [*untranslated-string*] (für nicht zu übersetzende Texte, umgeben von `'"'`), [*n*] (für Zahlen) und [*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 eincodiert, die Definitionen in der Spezifikationsdatei sind auszukommentieren.

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 $m = 3$ also genau 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 Interchange 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-nären Zahlen. Algorithmus 3 zeigt eine Lösung. Die Notation des Algorithmus verwendet das Entwurfsmuster 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.

```
1   procedure tuple iterator(m,k)
2     (t_1,...,t_m) := (1,...,1);
3     while t_1 ≤ k do
4       work with the tuple(t_1,...,t_m);
5       t_m := t_m + 1;
6       for i = m downto 2 do
7         if t_i > k then
8           t_i := 0;
9           t_{i-1} := t_{i-1} + 1;
10        else
11          exit for ;
```

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

$$\varkappa(m,k) = \sum_{i=0}^{m-1} k^i = \frac{k^{m+1}-k}{k-1} = O(k^m).$$

Da der Algorithmus zur Aufzählung der k^m Tupel jedes Tupel einmal anfassen muß, ist dies auch die untere Schranke für die Komplexität des Problem. 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 $t_i \le t_j$ $(i<j)$ beschränken. Algorithmus 3 ließe sich trivial adaptieren, indem man die ungewollten Tupel herausfiltert, somit also Zeile 4 durch

```
4A    if ∀ i < j : t_i ≤ t_j then
4B      work with the tuple(t_1,...,t_m);
```

ersetzt. Der resultierende Algorithmus behielte dann jedoch die gleiche Komplexität $O(k^m)$ und wäre nicht optimal, da man bloß an $O(\frac{k^m}{m!})$ 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 $\varkappa(m,k) = O(\frac{k^m}{m!})$. Dies ist wieder ein optimales Ergebnis.

```
1   procedure tuple iterator2(m,k)
2     (t_1,...,t_m) := (1,...,1);
3     while t_1 ≤ k do
4       work with the tuple(t_1,...,t_m);
5       t_m := t_m + 1;
6       for i = m downto 2 do
7         if t_i > k then
8           t_{i-1} := t_{i-1} + 1;
9         else
10          for j = i + 1 to m do
11            t_j := t_i;
12          exit for
```

Algorithmus 4 Aufzählen aller m-Tupel $(t_1,\ldots,t_m) \in \{1,\ldots,k\}^m$ modulo Permutation

Alle Möglichkeiten, Lotto zu spielen

Gesucht sind nun alle m-Tupel $(t_1,\ldots,t_m)$ mit $t_i \neq t_j$ $(i \neq j)$, 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 $(t_1,\ldots,t_m)$ mit $t_i \neq t_j$ $(i \neq j)$. 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 t_i 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 ersetzt, 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 $\binom{k}{m} = \frac{k!(k-m)!}{m!}$ Tupel

aufgezählt werden, benötigt der Algorithmus demnach also $O(\binom{k}{m})$ Schritte.

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

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 incl. Dokumentation und Quellen sowie die verwendeten Trainings- und Testmengen enthalten.

Es folgt eine Übersicht über die Materialen 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	Bilinguale Corpora Deutsch/Chinesisch
/nespole	Datenbanken von Nespole! in verschiedenen Formaten 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
/vogel	Statistischer Übersetzer von Stephan Vogel, vgl. Vogel and Ney (2000) und Abschnitt 7.3
/jtrans	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

/src	Quellcode
/talk	Folien zum Diplomarbeitsvortrag
/thesis	Die vorliegende Ausarbeitung

Symbolverzeichnis

$\mathrm{O}(f(n))$	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
$\mathbf{e} = (e_1, \ldots, e_i, \ldots, e_l)$	Wortfolge **e**, bestehend aus den l Wörtern e_1 bis e_l
$l = \|\mathbf{e}\|$	Länge des Satzes **e**
$\mathbf{f} = (f_1, \ldots, f_j, \ldots, f_m)$	Wortfolge **f**, bestehend aus den m Wörtern f_1 bis f_m
$m = \|\mathbf{f}\|$	Länge des Satzes **f**
$\mathbf{g} = (g_1, \ldots, g_k, \ldots, g_n)$	Wortfolge **g**, bestehend aus den m Wörtern g_1 bis g_n
$n = \|\mathbf{g}\|$	Länge des Satzes **g**
$\mathbf{a} = (a_1, \ldots, a_m)$	Alignment, Funktion, die jeder Quellsatzposition j eine Zieslsatzposition a_j zuordnet.
$t(f\|e)$	Übersetzungswahrscheinlichkeit, Wahrscheinlichkeit, daß f die Übersetzung des Wortes e ist
$\delta(x,y)$	Kroneckersymbol, 1, falls $x = y$ und 0 sonst.
$c(f\|e,\mathbf{f},\mathbf{e})$	Counts für die Übersetzungswahrscheinlichkeiten den IBM Modellen
$a(i\|j,m,l)$	Zuordnungswahrscheinlichkeit, Wahrscheinlichkeit dafür, daß $a_j = i$ gilt
$c(i\|j,m,l,\mathbf{e},\mathbf{f})$	Counts für die Zuordnungswahrscheinlichkeiten im IBM Modell 2
$n(\varphi_i\|e_i)$	Fertility-Wahrscheinlichkeit im IBM-3-Modell
$P(T)$	Perplexität einer Menge T
$PP(T)$	Phrasenperplexität der Menge T
$\partial\mathbf{t}$	Tiefe des Terms **t**
$\partial_\uparrow(s,\mathbf{t})$	Bottom-Up-Tiefe des Symbols s im Term **t**
$\partial_\downarrow(s,\mathbf{t})$	Top-Down-Tiefe des Symbols s im Term **t**
$L_\uparrow(i,\mathbf{t})$	Bottom-Up-Ebene i im Term **t**
$L_\downarrow(i,\mathbf{t})$	Top-Down-Ebene i im Term **t**
$\mathrm{top}_i(\mathbf{t})$	i-ter Kopf des Terms **t**
$\|A\|$	Mächtigkeit der Menge A
$\mathrm{wer}(\mathbf{t}_1,\mathbf{t}_2)$	Wortfehlerrate eines Baums $\mathbf{t}_1$ bzgl. $\mathbf{t}_2$
$[a,b]$	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
$\langle w \rangle$	Wortklasse w

$\mathscr{A}(f)$	Abbildung des Worts f in seine Quellsprach-Wortklasse
$\mathscr{B}(e)$	Abbildung des Worts e in seine Zielsprach-Wortklasse
$\varkappa(n)$	Komplexität eines Algorithmus in Abhängigkeit von n

Literatur

Balzert, H.: *Lehrbuch der Software-Technik*. Spektrum Akademischer Verlag, 2. Auflage. 2001.

Bangalore, S. and Riccardi, G.: A Finite-State Approach to Machine Translation. In *Proceedings of NAACL-2001*, Pittsburgh, USA. 2001.

Brown, P., Della-Pietra, S., Della-Pietra, V., and Mercer, R.: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, **19(2)**, 263–311. 1993.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C.: *Introduction to Algorithms*. MIT Press, 2. Auflage. 2001.

Han, B.: Task-Oriented Language Learning: Status Report. Technical report, Carnegie Mellon University, Pittsburg, USA. December 2002.

Hutchings, W.: Fifty Years of the Computer and Translation. *Machine Translation Review 6*, pages 22–24. October 1997.

IF. Nespole! Interchange Format Pages — Online Specification. May 2002. http://www.is.cs.cmu.edu/nespole/db/specification.html

Jelinek, F.: *Statistical Methods for Speech Recognition*. Massachusetts Institute of Technologie. 1999.

Knuth, D.: *The Art of Computer Programming*, volume 4, Combinatorical Algorithms. Addison Wesley. In preparation. 2002.

Langley, C., Lavie, A., Levin, L., Wallace, D., Gates, D., and Peterson, K.: Spoken Language Parsing Using Phrase-Level Grammars and Trainable Classifiers. To appear. 2002.

Lavie, A., Langley, C., Waibel, A., Pianesi, F., Lazzari, G., Coletti, P., Taddei, L., and Balducci, F.: Architecture and Design Considerations in Nespole!: a Speech Translation System for e-Commerce Applications. In *Proceedings of HLT-2001*, pages 31–34, San Diego, USA. March 2001.

Lee, Y., Yi, W., Seneff, S., and Weinstein, C.: Interlingua-Based Broad-Coverage Korean-to-English Translation in CCLINC. In *Proceedings of HLT-2001*, pages 194–199, San Diego, USA. March 2001.

Levin, L., Gates, D., Lavie, A., and Waibel, A.: An Interlingua Based on Domain Actions for Machine Translation of Task-Oriented Dialogues. In *Proceedings of ICSLP-1998*, pages 1155–1158, Sydney, Australia. November 1998.

Levin, L., Gates, D., Lavie, A., Pianesi, F., Wallace, D., Watanabe, T., and Woszczyna, M.: Evaluation of a Practical Interlingua for Task-Oriented Dialogue. In *Workshop of the SIG-IL, NAACL*, pages 18–23, Seattle, USA. May 2000.

Levin, L., Bartlog, B., Llitjos, A., Gates, D., Lavie, A., Wallace, D., Watanabe, T., and

Woszczyna, M.: Lessons Learned from a Task-Based Evaluation of Speech-to-Speech Machine Translation. In *Proceedings of LREC-2000*, pages 721–724, Athens, Greece. 2000.

Linke, A., Nussbaumer, M., and Portmann, P.: *Studienbuch Linguistik*. Niemeyer, 3. Auflage. 1996.

Macherey, K., Och, F., and Ney, H.: Natural Language Understanding Using Statistical Machine Translation. In *Proceedings of Eurospeech 2001*, pages 2205–2208, Aalborg, Denmark. September 2001.

Nasr, A., Rambow, O., Palmer, M., and Rosenzweig, J.: Enriching Lexical Transfer With Cross-Linguistic Semantic Features or How to Do Interlingua Without Interlingua. In *Proceedings of the Second International Workshop on Interlingua*, San Diego, USA. October 1997.

Ney, H., Och, F., and Vogel, S.: Statistical Translation of Spoken Dialogues in the Verbmobil System. In *Proceedings of Workshop on Multi-Lingual Speech Communication*, pages 69–74, Kyoto, Japan. October 2000.

Och, F.: An Efficient Method for Determining Bilingual Word Classes. In *Proceedings of EACL-1999*, pages 71–76, Bergen, Norway. June 1999.

Och, F., Tillmann, C., and Ney, H.: Improved Alignment Models for Statistical Machine Translation. In *Proceedings of EMNLP/VLC-1999*, pages 20–28, University of Maryland, USA. June 1999.

Vogel, S. and Ney, H.: Translation with Cascaded Finite State Transducers. In *Proceedings of ACL-2000*, pages 23–30, Hongkong, China. October 2000.

Vogel, S. and Tribble, A.: Improving Statistical Machine Translation for a Speech-to-Speech Translation Task. To appear. 2002.

Vogel, S., Ney, H., and Tillmann, C.: HMM-Based Word Alignment in Statistical Translation. In *Proceedings of ICCL-1996*, pages 836–841, Kopenhagen, Denmark. August 1996.

Vogel, S., Och, F., Tillmann, C., Nießen, S., Sawaf, H., and Ney, H.: Statistical Methods for Machine Translation. *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 377–393. 2000.

Wang, Y.: *Grammar Inference and Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, USA. 1998.

Wang, Y. and Waibel, A.: Decoding Algorithm in Statistical Machine Translation. In *Proceedings of the EACL-1997*, pages 366–372, Madrid, Spain. 1997.

Index

K