Representation and stochastic resolution of ambiguity in constraint-based parsing

Darstellung und stochastische Auflösung von Ambiguität in constraint-basiertem Parsing

Von der philosophischen Fakultät der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil)
genehmigte Dissertation

von

Andreas Eisele,
Meylan

Hauptberichter: Prof. Dr. Christian Rohrer
Mitberichter: Prof. Dr. Mats Rooth
Tag der mündlichen Prüfung: 9. Februar 1999

Hiermit erkläre ich, daß ich die vorliegende Dissertation eigenständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Meylan, den 20. Januar 1999

_____

(Andreas Eisele)

Contents

Zusammenfassung

Ambiguität in der Analyse natürlicher Sprache

Die natürliche Sprache spiegelt die menschlichen intellektuellen Fähigkeiten wider, die bei weitem nicht voll verstanden sind. Solange wir die Prozesse, die bei Denken und Verstehen ablaufen, nicht formal beschreiben können, müssen auch die formalen Beschreibungen des menschlichen Sprachverhaltens grobe Approximationen bleiben. Ein markantes Beispiel für diesen generellen Sachverhalt ist das Problem der Disambiguierung sprachlicher Äußerungen. Da unser Gebrauch von Worten unseren Fähigkeiten entspricht, deren Sinn, Kontext und Absicht zu verstehen, können Systeme, denen solche Fähigkeiten abgehen, bestenfalls Mengen möglicher Analysen produzieren. Es ist wohlbekannt, daß solche Mengen in der Praxis sehr groß werden können. Mehrdeutigkeit ist eines der Grundprobleme in der automatischen Sprachverarbeitung, und jedes sprachverarbeitende System benötigt Methoden, damit umzugehen.

Mehrdeutigkeiten existieren auf vielen linguistischen Ebenen. Wir können grob lexikalische und strukturelle Ambiguität unterscheiden, aber dies sind wiederum Sammelbegriffe für eine Vielzahl unterschiedlicher Probleme. Lexikalische Mehrdeutigkeit kann die Wortart eines gegebenen Wortes betreffen, seine morphologischen Eigenschaften, seine Subkategorisierung (also die Anzahl und der Typ der Komplemente, mit denen dieses Wort auftritt), seine Bedeutung oder jegliche sonstige Information, die diesem Wort zugeordnet wird.

[Bar-Hillel1960] gab das Beispiel *"The box is in the pen"* um zu

zeigen, daß die Bestimmung der Wortbedeutung[1] ein sehr schweres Problem darstellt, zu dessen Auflösung in manchen Fällen eine unbegrenzte Menge an Weltwissen notwendig ist. Das Problem der Disambiguierung der Wortbedeutung wurde daher als "KI-vollständig" charakterisiert. Ein ausführlicher Überblick über den Stand der Forschung findet sich in [Ide and Véronis1998].

Strukturelle Mehrdeutigkeit ist Mehrdeutigkeit in der Zusammenwirkung mehrerer Worte. Sie tritt auf, wenn Worte auf unterschiedliche Weisen gruppiert werden können, oder wenn es verschiedene Interpretationen gibt, in welcher Relation Worte oder größere Konstituenten stehen. Diese Unterschiede können die Anbindung bestimmter Phrasen betreffen, den Skopus von Quantifikatoren oder den Bezug von anaphorischen Ausdrücken. Dies ist nur ein sehr kleiner Ausschnitt der relevanten Phänomene. Ein sehr berühmtes Beispiel ist der Satz *"I saw the man on the hill with the telescope."* (Ich sah/säge den Mann auf dem Berg mit dem Fernrohr), der etliche unterschiedliche Interpretationen hat, die sich – neben der lexikalischen Ambiguität des Verbes – in der Zuordnung der Präpositionalphrasen *"on the hill"* und *"with the telescope"* unterscheiden.

Die Unterscheidung zwischen lexikalischer und struktureller Ambiguität ist jedoch nicht ganz eindeutig. Wenn wir beispielsweise entscheiden müssen, ob ein Paar aufeinanderfolgender Worte eine lexikalische Einheit bilden oder ob sich die Bedeutung dieser Wortgruppe kompositionell aus der Bedeutung der einzelnen Worte herleiten läßt, dann ist es eine Frage der Definition, diese Art von Mehrdeutigkeit als lexikalische oder strukturelle Ambiguität oder als beides zu klassifizieren.

Im Allgemeinen interagieren Mehrdeutigkeiten auf verschiedenen Ebenen in dem Sinne, daß die Festlegung bestimmter möglicher Entscheidungen (z.B. die Wortart eines Wortes) auch die Zahl der Möglichkeiten in einigen anderen Entscheidungen verringert. In dem ebenfalls sehr bekannten Beispiel *"Time flies like an arrow."* kann *flies* entweder Verb (= fliegt) oder Nomen (= Fliegen) sein, und diese lexikalische Mehrdeutigkeit interagiert mit den verschiedenen möglichen Bedeutungen von *time* und mit der Mehrdeutigkeit, die die gesamte syntaktische Struktur dieses Satzes betrifft.

[1] "Pen" steht hier für einen Laufstall.

In vielen praktischen Anwendungen stellt die Art und Weise, in der Äußerungen in das sprachverarbeitende System eingegeben werden, eine weitere Quelle von Unsicherheit dar. Sehr oft weichen die eingegebenen Daten vom Ideal einer wohlgeformten natürlichsprachlichen Äußerung ab. Systeme, die textuelle Eingabe verarbeiten, müssen sehr oft mit Tipp- oder Schreibfehlern klarkommen. Text, der durch ein OCR-System automatisch digitalisiert wurde, enthält typischerweise mehrere Fehler pro Seite. Die Situation verschlimmert sich, wenn OCR auf Handschrift angewandt wird, oder wenn ein sprachverarbeitendes System die Ausgabe maschineller Spracherkennung verwenden soll. Zwar gibt es in diesen Situationen Wege, die Qualität der Daten automatisch zu verbessern, diese Techniken können jedoch nicht unabhängig von der linguistischen Verarbeitung eingesetzt werden, da sie Zugang zu linguistischen Wissensquellen und zum Kontext der Äußerungen benötigen. Die Korrektur fehlerhafter Eingaben führt oft nicht zu einem eindeutigen Resultat und verschlimmert daher das Problem der Mehrdeutigkeit. Im Falle der maschinellen Spracherkennung ist es offensichtlich, daß die Abbildung von Lauten nach Worten nicht eindeutig sein kann, da verschiedene Worte gleich klingen können. Jedes System, das gesprochene Eingaben in deren korrekte Schreibweise überführen soll, muß daher in solchen Fällen Wissen über den Kontext heranziehen. Da Mehrdeutigkeiten gleichzeitig auf mehreren Ebenen in derselben Äußerung auftreten können, kann sich die Zahl der Möglichkeiten zu einer enormen Anzahl an Kombinationen multiplizieren.

Ein erstaunlicher Aspekt des Problems ist die Tatsache, daß Menschen die Mehrdeutigkeiten ihrer Äußerungen normalerweise nicht bemerken. In einer typischen kommunikativen Situation versucht der Sprecher sich so auszudrücken, daß die Adressaten die Äußerung "kapieren". Grice hat das kooperative Prinzip der Kommunikation formuliert, und unter den vier Anforderungen, die er unter dem Axiom der Art und Weise aufführt, nennt er ausdrücklich "Vermeide Mehrdeutigkeit" [Levinson1983]. Das genaue Ergebnis davon hängt von den Annahmen ab, die der Sprecher über das Wissen des Addressaten macht. Wer jemals versucht hat, einen technischen oder wissenschaftlichen Fachtext eines ihm fremden Bereichs zu verstehen, wird wissen, daß selbst die einfachsten Schritte der syntaktischen Analyse

fehlschlagen können, wenn die Bedeutung mancher Worte nicht bekannt ist. Mehrdeutigkeit ist daher das Problem fehlenden Wissens auf seiten eines verarbeitenden Systems, für das eine Äußerung ursprünglich nicht bestimmt war.

Menschliche Disambiguierung benutzt viele unterschiedliche Wissensquellen, die jedoch auf nahtlose Weise zusammenarbeiten. Die linguistische Forschung hat andererseits stets gut daran getan, die Regelmäßigkeiten auf verschiedenen sprachlichen Ebenen getrennt zu beschreiben, und unterscheidet deshalb beispielsweise Morphologie (Wortbildung), Syntax, Semantik und die Modellierung domänenspezifischen ontologischen Wissens.

Für diese Ebenen sind verschiedenen Arten von Wissen relevant, zu deren Beschreibung unterschiedliche Formalismen geeignet sind, für die die maschinelle Sprachverarbeitung unterschiedliche Verarbeitungsstrategien entwickelt hat. In manchen dieser Ebenen können die beobachteten Regelmäßigkeiten am besten mittels symbolischer Regeln formuliert werden, die die Menge der Strukturen, die in Betracht kommen, klaren Bedingungen unterwerfen. Allerdings beschränken sich diese regelbasierten symbolischen Beschreibungen auf ja-nein Unterscheidungen; sie können daher graduelle Unterschiede in der Annehmbarkeit von Äußerungen oder Interpretationen nicht ausdrücken. Sie werden typischerweise zur Charakterisierung der menschlichen *Sprachkompetenz* benutzt, d.h. im Rahmen einer Theorie aller wohlgeformten Äußerungen und aller möglichen Interpretationen derselben. Eine Theorie der Kompetenz alleine kann uns nicht sagen, wie Mehrdeutigkeiten aufzulösen sind. Wenn eine Äußerung mehrere mögliche Interpretationen hat, gibt sie keinen Hinweis, wie zwischen ihnen entschieden werden kann.

Wenn wir uns jedoch der menschlichen *Sprachperformanz* zuwenden, also der Frage, wie der unendlich Vorrat potentieller Äußerungen praktisch genutzt wird, so finden wir viele weitere relevante Wissensquellen. Beobachten wir, wie grammatikalische Konstruktionen im menschlichen Diskurs tatsächlich verwendet werden, so finden wir weiteren Regelmäßigkeiten, wie beispielsweise die relative Häufigkeit solcher Konstruktionen, Beschränkung der Verschachtelungstiefe bei Konstruktionen mit zentraler Einbettung, oder die Tendenz, daß koordinierte Phrasen sich in Struktur und Komplexität ähneln.

Aber wir können Information über die Häufigkeit einer Konstruktion nicht einfach in eine Beurteilung der Annehmbarkeit einer Äußerung umsetzen. Es gibt beispielsweise keine klare Grenze für die Verschachtelungstiefe, und der tatsächliche Gebrauch hängt von vielen Faktoren wie Textsorte und persönlichem Stil ab. Die Tendenz zur Parallelität in der Koordination kann nicht in eine Menge exakter Bedingungen übersetzt werden, die den Fakten gerecht wird. Ähnliche Beobachtungen können hinsichtlich der partiell freien Anordnung von Konstituenten in vielen Sprachen gemacht werden [Uszkoreit1984].

Untersuchen wir Phänomene der Wortwahl, indem wir statistische Information über das gemeinsame Vorkommen von Worten sammeln, so können wir ebenfalls starke Tendenzen und Muster auffinden, die sich nicht in harte Fakten übersetzen lassen. Die Muster hängen nicht nur von Textsorte und Stil ab, sondern auch in ganz starkem Maße vom Gegenstandsbereich des Diskurses. Durch diese Vielfalt wird jegliche theoretische Betrachtung erheblich erschwert.

Bei genauerer Betrachtung können wir zwei Quellen "weicher Constraints" unterscheiden. Zum einen sind dies linguistische Präferenzen, die eine von mehreren Möglichkeiten, einen bestimmten Inhalt auszudrücken, wahrscheinlicher machen als die anderen. Diese Präferenzen betreffen Wortwahl und linguistische Konstruktionen, und sie hängen von Textsorte und Stil ab. Die andere Quelle weicher Constraints hängt mit den Inhalten zusammen, die in den Äußerungen beschrieben werden. Bestimmte Inhalte sind wahrscheinlicher als andere; wenn also eine Äußerung so verstanden werden kann, daß sie mit unterschiedlichen Wahrscheinlichkeiten verschiedene Inhalte beschreiben könnte, dann werden wir die wahrscheinlichste Interpretation bevorzugen[2]. Natürlich hängen solche Wahrscheinlichkeiten vom Gegenstandsbereich des Diskurses ab, aber auch von vielen weiteren Eigenschaften der Situation, in der eine Äußerung vollzogen wird. Die Schätzung von Wahrscheinlichkeiten liegt außerhalb der Möglichkeiten der Linguistik, aber offensichtlich ist solches Wissen notwendig, um die menschliche Fähigkeit zum Umgang mit Sprache im Allgemeinen und

[2]Dies entspricht nicht notwendigerweise der wahrscheinlichsten Tatsache. Wir müssen pragmatische Gesichtspunkte wie das kooperative Prinzip von Grice (siehe [Levinson1983]) berücksichtigen, um zu bewerten, welche Interpretationen in einer bestimmten Dialogsituation wahrscheinlich sind.

zur Disambiguierung im Besonderen simulieren zu können.

Werden die verschiedenen Ebenen der linguistischen Theorie implementiert und zum Bau eines sprachverarbeitenden Systems verwendet, so mag es notwendig erscheinen, alle diese Ebenen in einer großen Maschinerie zu kombinieren, die alles verfügbare Wissen verwendet und alle Teilprobleme in einem Durchlauf erledigt. Aber ein solches integriertes System würde die Modularität verlieren und wäre daher sehr schwer zu verwalten. Deshalb werden in einer traditionellen Architektur die verschiedenen linguistischen Ebenen als Module einer sequentiellen, mehrphasigen Verarbeitungsarchitektur realisiert, die spezialisierte Prozessoren für jede Ebene vorsieht, durch die die Eingabe durchgereicht wird. Allerdings ist es in einer solchen Architektur schwierig, die Interaktion der vielen Wissensquellen zu modelieren, die auf den unterschiedlichen Ebenen operieren.

Wenn wir verlangen, daß jede dieser Komponenten eine einzige und eindeutige Repräsentation ihrer Ausgabe an die nächste Komponente abliefert, so muß sie Entscheidungen treffen, für die sie nicht genügend Information besitzt. Es ist wichtig, der Versuchung zu widerstehen, Mehrdeutigkeiten sofort (ad hoc) aufzulösen, sobald sie auftreten, denn eine lokale Analyse, die von irgend einem Verarbeitungsmodul vorzeitig verworfen wird, könnte für eine korrekte Behandlung der gesamten Äußerung in einer späteren Komponente wesentlich sein.

Eine naheliegende Lösung für Probleme dieser Art besteht darin, Mengen von Analysen zwischen den Komponenten auszutauschen, die in späteren Verarbeitungsschritten verwendet oder aber verworfen werden können. Das praktische Problem hierbei ist die Vielzahl der Ambiguitäten, insbesondere die Tatsache, daß diese oft unabhängig voneinander sind, so daß sie in ihrer Gesamtheit zu einem exponentiellen Anwachsen der Zahl der Lösungen führen. Bereits eine Unsicherheit in der Interpretation eines einzelnen Wortes kann die Anzahl der Interpretationen der gesamten Äußerung verdoppeln. Lösungen in solchen Mengen stimmen oft in großen Teilen der in ihnen enthaltenen Information überein. Wenn die jeweils nächste Komponente in der Verarbeitungskette gezwungen ist, alle Lösungen einzeln zu betrachten, so wird sie solche Gemeinsamkeiten nicht ausnützen können und eine exponentielle Menge an Verarbeitungsressourcen verbrauchen.

Zwei Wege zum Umgang mit Ambiguität

Die vorliegende Arbeit untersucht Techniken, die linguistische und statistische Wissensquellen kombinieren, um syntaktische und lexikalische Ambiguität zu repräsentieren und aufzulösen.

Die Vorgehensweise beruht auf einer constraintbasierten Grammatik, die als einfache Formalisierung der syntaktischen Regelmäßigkeiten der vorliegenden Sprache dient. Durch den Gebrauch einer solchen Grammatik können wir die Aufmerksamkeit auf syntaktisch wohlgeformte Äußerungen konzentrieren und solche Äußerungen in Mengen zugrundeliegender Strukturen abbilden, die typischerweise syntaktische und etwas semantische Information enthalten. Ich werde versuchen, die Methode – so weit wie möglich – unabhängig von Einzelheiten des Formalismus und der Grammatik zu halten. In Fällen, in denen Einzelheiten notwendig sind, werde ich Beispiele auf der Grundlage von LFG [Kaplan and Bresnan1982] oder DCG [Pereira and Warren1980] verwenden. Ich werde annehmen, daß die Strukturen, die von der Grammatik aufgebaut werden, eine implizite oder explizite Repräsentation der zugrundeliegenden Prädikat-Argument-Struktur enthalten, d.h. daß sie festlegen, welche Einheiten in welcher Art von Handlungen verwickelt sind.

In diesem Rahmen betrifft Ambiguität alle Situationen, in denen einer einzelnen Äußerung mehr als eine solche Struktur zugeordnet wird. Die genaue Auswahl an Phänomenen, die durch diese generische Definition abgedeckt sind, hängt von den Einzelheiten der Grammatik ab. Wir können eine Grammatik schreiben, die die Worte in die Konzepte einer semantischen Ontologie abbildet. In diesem Fall werden für sehr viele Worte ambige Analysen entstehen, und die Aufgabe, für jedes Wort das richtige Konzept zu identifizieren, ist sehr viel schwieriger als in Situationen, in denen die Menge der Prädikate weitgehend durch die Worte bestimmt wird, die betroffen sind. Die vorliegende Arbeit versucht nicht, im Detail die verschiedenen Formen der Ambiguität zu analysieren und für jede von ihnen eine spezielle Methode zur Disambiguierung anzugeben. Sie zielt vielmehr auf ein allgemeines Schema, das je nach Bedarf auf viele Formen der Ambiguität angewandt werden kann.

Im ersten Teil, der aus den Kapiteln 2 und 3 besteht, skizziere

ich Wege, die Disambiguierung zu vermeiden, indem mehrere mögliche Interpretationen in einer kompakten Form representiert werden. Die darin beschriebene Arbeit basiert auf Ideen, die maßgeblich von Ron Kaplan, Martin Kay und John Maxwell im Zusammenhang mit Implementierungen von LFG entwickelt wurden. Vergleichbare Ansätze wurden auch im Zusammenhang mit Logik-basierter Sprachverarbeitung vorgestellt. Es ist allerdings nützlich, die verschiedenen Methoden für gepackte Repräsentationen von Lösungsmengen in einer einheitlichen Form darzustellen, um dabei einige zugrundeliegenden gemeinsame Prinzipien zu beleuchten.

Im zweiten Teil stelle ich eine Methode zur partiellen Auflösung von Ambiguität mittels stochastischer Modelle vor. Das Ziel der Disambiguierung wird hierbei nicht in erster Linie darin gesehen, eine einzelne Lösung aus der Menge der möglichen Lösungen auszuwählen, sondern darin, jeder Analyse eine Bewertung in Form einer Wahrscheinlichkeit zuzuordnen. Diese Bewertung führt in naheliegender Weise zur Auswahl einer "besten" Lösung, falls eine solche Auswahl erforderlich sein sollte. Eine probabilistische Bewertung ist jedoch sehr viel mächtiger. Sie definiert eine Rangordnung, d.h. eine nach Wahrscheinlichkeit sortierte Aufzählung der Lösungen. Anhand einer solchen Rangordnung können wir ermitteln, ob zwischen der ersten und zweiten Lösung ein großer Unterschied besteht, was nützlich bei der Entscheidung sein kann, ob die erste Lösung ohne weitere Prüfung ausgewählt werden soll, oder ob weitere (möglicherweise aufwendigere) Mechanismen zur Auswahl herangezogen werden sollen. Von einer Wahrscheinlichkeitsverteilung über Lösungen können wir außerdem die Wahrscheinlichkeiten bestimmter Eigenschaften der Lösungen ermitteln, was zur Sammlung von zusätzlichem statistischem Trainingsmaterial dienen kann, oder dazu, die interaktive Disambiguierung möglichst effizient zu gestalten.

Ein weiterer Vorteil dieser probabilistischen Bewertung kommt zum Vorschein, wenn wir die Analyse als Eingabe weiterer Verarbeitungschritte verwenden wollen, beispielsweise zum Transfer in eine andere Sprache, oder um in einem Dialogsystem bestimmte Handlungen durchzuführen. Die nachfolgende Komponente kann möglicherweise nicht jede Analyse akzeptieren, die von der syntaktischen Analyse produziert wird, oder sie könnte bestimmte Eingaben anderen vorziehen.

In beiden Situationen bietet die Existenz einer stochastisch bewerteten Sammlung von Möglichkeiten einen guten Weg, weiche oder harte Constraints beider Komponenten zu verbinden, um die insgesamt beste Lösung zu finden.

Die Integration grammatikalischer und statistischer Ansätze zur Sprachverarbeitung ist ein Gebiet, in dem zur Zeit intensiv geforscht wird, wobei noch keine abschließenden Ergebnisse gefunden wurden. Ich versuche in meiner Arbeit einige Bausteine anzubieten, die zu einer Lösung beitragen könnten. Die volle Integration in ein sprachverarbeitendes System und der Vergleich mit anderen Techniken, die in der mittlerweile sehr umfangreichen Literatur vorgeschlagen wurden, muß jedoch zukünftiger Arbeit vorbehalten werden.

Während die Aufgabe der Disambiguierung zwischen konkurrierenden syntaktischen Analysen die Hauptmotivation dieser Arbeit darstellt, bestehen auch starke Querbezüge zu Arbeiten zur stochastischen Sprachmodellierung, also zu Ansätzen, die Wahrscheinlichkeitverteilungen über mögliche Äußerungen definieren. Einerseits verwende ich einige Techniken, die für die Sprachmodellierung entwickelt wurden und verallgemeinere diese, um sie für Verteilungen auf syntaktischen Dependenzstrukturen verwenden zu können. Andererseits stellen die so definierten Modelle implizit auch stochastische Modelle der Zeichenketten dar, mit denen diese Dependenzstrukturen assoziiert sind. Dabei ist zu hoffen, daß durch Ausnutzung syntaktischer Abhängigkeiten neue und bessere Sprachmodelle geschaffen werden können, die prinzipiell die Genauigkeit von Spracherkennung, OCR oder anderen Anwendungen erhöhen können, bei denen eine beeinträchtigte oder ambige Repräsentation natürlicher Sprache dekodiert werden muß.

Aufbau der Arbeit

Die Arbeit ist folgendermaßen gegliedert.

Im Kapitel 2 bespreche ich einige untereinander verwandte Vorschläge zur Darstellung disjunktiver Information bei der Analyse mittels constraintbasierter Grammatiken. Kapitel 3 untersucht die Frage, wie ambige Ergebnisse beim kontextfreien Parsing kompakt

repräsentiert werden können, und wie die Schemata zur Repräsentation disjunktiver Merkmalsinformation in die Repräsentation von Parse-Wäldern integriert werden können.

Kapitel 4 gibt eine informelle Einführung in die Verwendung von stochastischen Modellen in der Sprachverarbeitung im Allgemeinen, und zu Disambiguierung im Besonderen. In Kapitel 5 skizziere ich Wege, die Wahrscheinlichkeit von Worten zu schätzen, wobei der Kontext ignoriert wird. Obwohl diese Fragestellung per se nicht sehr interessant oder wichtig erscheint, erlaubt sie doch einige Einsichten in das allgemeinere Schätzproblem. Ich untersuche auch, wie mittels morphologischer Information die Schätzung von Wortwahrscheinlichkeiten verbessert werden kann und wie lexikalische Information und Trainingsdaten von anderen Domänen ausgenutzt werden können.

Kapitel 6 behandelt einfache Lösungen zum Problem, bilexikalische Wahrscheinlichkeiten zu schätzen, d.h. die Wahrscheinlichkeiten, daß ein Wortpaar in einer bestimmten syntaktischen Relation auftritt. Ich untersuche Methoden, die auf einer Mischung von kontextsensitiven Verteilungen und Randverteilungen basieren. Ich beschreibe, wie die in Kapitel 5 entwickelte Technik zur Kombination spezieller und allgemeiner Evidenz auf Häufigkeiten von Bigrammen verallgemeinert werden kann, was zu einer neuartigen Schätzmethode führt. Das Kapitel untersucht auch, wie Ähnlichkeiten der Verteilung zwischen Worten ausgenützt werden können, um verbesserte Schätzungen von Bigram-Wahrscheinlichkeiten zu erhalten. Ich gebe einen Überblick über eine größere Zahl von Techniken, die bisher für dieses Problem verwendet wurden, die von harter Gruppierung (clustering) über weiche Gruppierung bis hin zu Ansätzen reichen, die die nächsten Nachbarn zur Schätzung heranziehen. Ich diskutiere einige Alternativen bei der Wahl geeigneter Ähnlichkeitmaße und skizziere, wie Probleme mit der Platz- und Zeitkomplexität dieser Verfahren gelöst werden können. Kapitel 7 diskutiert die Frage, wie mehrere Quellen kontextueller Evidenz kombiniert werden können, ohne daß dies zu einem exponentiellen Anwachsen der zu schätzenden Parametern führt. Ein Weg, stochastische Modelle stärker kontextsensitiv zu machen, besteht darin, sie direkt auf eine Beschreibung in einer ausdrucksstärkeren Sprache aufzubauen, wie z.B. auf eine constraintbasierte Grammatik. Dadurch wird es prinzipiell möglich, beliebige strukturelle Eigenschaften in die Bewertungsfunk-

tion aufzunehmen. Allerdings erschwert dieser allgemeine Ansatz die Schätzung der Parameter. Kapitel 8 beschreibt ein Experiment, in dem viele der besprochenen Techniken gemeinsam verwendet werden, um die mit einer großen lexikalisch funktionalen Grammatik erhaltenen Parse-Wälder zu disambiguieren. Schließlich enthält Kapitel 9 eine kurze Zusammenfassung und beschreibt mögliche Richtungen zukünftiger Arbeit.

Chapter 1

Introduction

1.1 Ambiguity in Natural Language Processing

Natural language mirrors the human intellectual capacities, which are far from being understood. As long as we can not formally describe the processes involved in thinking and understanding, formal descriptions of human language behavior have to be rough approximations. One particular instance of this general fact is the problem of disambiguation of human utterances. Since our use of words fits our capabilities of understanding their meaning, context and intent, systems that do not have such capabilities can, at best, produce sets of possible analyses. It is well known that such sets can be very large in practice. Ambiguity is hence one of the basic problems in automatic processing of natural language, and any natural language processing (NLP) system needs ways to cope with it.

Ambiguity exists on many linguistic levels. Roughly, we can distinguish lexical and structural ambiguities, but each of them is again a collective term for rather diverse sets of problems. *Lexical ambiguity* can affect the part of speech of a given word, its morphological features, its subcategorization, i.e. the number and types of the complements the word comes with, its meaning, or any other information assigned to it. [Bar-Hillel1960] gave the example *"The box is in the pen"* to show

that determining the proper sense of a word[1] is a very hard problem that, in some cases, requires an unlimited amount of world knowledge to be resolved. The problem of word sense disambiguation has therefore been described as "AI-complete". An extensive survey of the state of the art in word sense disambiguation is given in [Ide and Véronis1998].

Structural ambiguity is ambiguity in the way in which words interact. It comes up if there are different possible groupings of the words or different interpretations of the interrelations between words or larger constituents. These differences can affect the attachment of certain phrases, the scope of quantifiers, and the reference of anaphora, among many other phenomena. One very famous example is the sentence *"I saw the man on the hill with the telescope."* which has several possible interpretations, differing in the attachment of the prepositional phrases *"on the hill"* and *"with the telescope"*.

However, the distinction between lexical and structural ambiguity is not clear-cut. For instance, if we have to decide if a pair of adjacent words form a lexical unit or if the meaning of this word group can be inferred from the individual meanings in a compositional way, it is a matter of definition to classify this kind of ambiguity as lexical or structural or both.

More generally, ambiguity on different levels interact in the sense that fixing some of the possible decisions, such as the choice of the part of speech of a word, can also reduce the number of possibilities in some other choices, like the possible attachments of a phrase. In the also very well known example *"Time flies like an arrow."*, *flies* can be verb or noun, and this lexical ambiguity interacts with the ambiguity concerning the overall syntactic structure of this sentence.

In many practical applications, the way utterances are entered into the linguistic processing system is a source of additional uncertainty. Very often, the input data deviates from the ideal of well-formed natural language utterances. Systems that process textual input often need to cope with typing or spelling errors. Text that has been automatically digitized by OCR typically contains several errors per page. The situation is worse if OCR has been applied to hand writing or if NLP is to work on the output of a speech recognition system. Although there

[1] *pen* in this case means an enclosure with a fence, or a play-pen.

are some ways to automatically improve the quality of the data under such circumstances, these techniques cannot be applied separately from linguistic processing, since they need access to linguistic knowledge sources and the context of the utterance. Correction of ill-formed input typically does not lead to a unique result and hence adds to the ambiguity problem. In the case of speech recognition, it is quite obvious that the mapping from sounds to words cannot be unique, since different words may sound alike. Any system that needs to restore the correct orthography from spoken input will have to use contextual knowledge in these cases. Since ambiguity can simultaneously appear on several levels in the same utterance, the number of possibilities can multiply to an enormous space of combinations.

A striking fact about the problem is that humans usually don't even perceive the ambiguity of the utterances they use. In typical communicative situations, speakers try to express themselves in such a way that that their intended audience will "get the message". Grice has stated the co-operative principle of communication, and among the four requirements listed under the maxim of manner, he lists "avoid ambiguity" [Levinson1983]. The exact result of this depends on the speaker's assumption about the knowledge of the audience. Anyone who has ever tried to understand technical or scientific text from an unfamiliar domain will know that even basic steps of parsing can fail if the meaning of some of the words is unknown. Hence ambiguity is mainly a problem of missing knowledge on the side of the system for which some utterance was not intended.

Human disambiguation makes use of many knowledge sources of very different nature, and they seem to interact in a seamless way. Linguistic research, on the other hand, has always, and for good reasons, distinguished several relevant levels, such as morphology, syntax, semantics, and the modeling of domain-dependent ontological knowledge.

For these levels, different kinds of knowledge and different description formalisms seem most appropriate, and computational linguistics has used different processing strategies for them. For some of these levels, the observed regularities are most naturally described using symbolic rules, which impose hard constraints on the set of structures that should be taken into consideration. However, rule-based symbolic de-

scriptions are limited to yes-no distinctions and cannot express gradual differences in the acceptability of utterances or interpretations. They are typically used for a characterization of human language *competence*, i.e. a theory of all well-formed utterances and their possible interpretations. A theory of competence alone cannot tell us how to resolve ambiguity. If an utterance has several possible interpretations, there is no way to differentiate between them.

However, if we turn our attention to human language *performance*, we find many additional potential knowledge sources. By observing the way grammatical constructions are actually used in human discourse, we can find more regularities, such as the relative frequency of these constructions, limitations in the depth of center embedding recursion, or the tendency of coordinated phrases to be similar in structure and complexity. But we cannot simply turn information on the frequency of a construction into a judgment of the acceptability of an utterance. For instance, there is no hard limit on the number of recursive embeddings, and actual usage depends on many factors such as text genre and personal style. There is no way to translate the tendency of parallelism in coordination into a set of hard constraints on the structures that would adequately describe the facts. Similar observations can be made with respect to partially free order of constituents in many languages[Uszkoreit1984].

If we start to study matters of lexical choice by collecting statistical information on word co-occurrences, we can again find quite strong tendencies and patterns which yet cannot be translated into hard facts. The patterns depend not only on genre and style, but also quite heavily on the domain of discourse. This variety makes any theoretical investigation considerably more complex.

If we look more closely, we can distinguish two sources of soft constraints. One is *linguistic* preferences, that make one of several possible ways to express some given meaning more likely than another. These preferences affect words and linguistic constructions, and they depend on genre and style. The other source of soft constraints has to do with the meanings that are described in the utterances. Some meanings are more likely than others, hence if some utterance could be understood as describing different meanings with different probabilities, we will prefer

the most likely interpretation[2]. Of course, such probabilities depend on the domain of the discourse, but also on many more properties of the situation in which an utterance is made. Estimating these probabilities in a given situation is beyond the scope of linguistics, but obviously such knowledge is important for simulating the human language processing capabilities in general and disambiguation behavior in particular.

When the different levels of linguistic theory are implemented and used to build an NLP system, one might feel the need to integrate all these levels into one big machinery that uses all available knowledge and does everything in one pass. But such an integrated system would lose its modularity and become quite hard to manage. Hence a traditional architecture will translate the linguistic levels into modules of a sequential, multi stage processing architecture with specialized processors for each of the levels, through which the input is passed. But in such an architecture, it is difficult to model the interaction of many knowledge sources that operate on different levels.

If we force each module to deliver a single and unambiguous representation of its output to the next component, it must make decisions for which it does not have sufficient information. It is important to resist the temptation to resolve ambiguities as soon as they appear, in an ad-hoc manner, since a local analysis that is prematurely rejected by some processing module may be essential for a correct treatment of the overall utterance in some later component.

A straightforward solution to these kinds of problems is to pass sets of analyses between the components, which may be filtered in subsequent processing steps. However, the practical problem here is that there are so many ambiguities, that they are often independent, and that their totality may lead to an exponential increase in the number of solutions. An uncertainty in the interpretation of a single word may already double the number of interpretations of the whole utterance. Solutions in such sets often have large proportions of their information in common. If the next component in the chain is forced to look at all the solutions in isolation, it will not be able to exploit common parts of

[2]This is not necessarily the interpretation that describes the most probable fact. We have to take pragmatic constraints such as Grice's co-operative principle (see [Levinson1983]) into account to assess which interpretations are probable in a certain dialog situation.

the structures and will require an exponential amount of computational resources.

1.2 Two Ways to Cope with Ambiguity

This thesis investigates techniques that combine linguistic and statistical knowledge sources for the representation and resolution of syntactical and lexical ambiguities. It tries to integrate two complementary approaches to deal with the problem.

The basic setup relies on a constraint-based grammar which is seen as a simple formalization of the syntactic regularities of the given language. By using such a grammar, we are able to restrict attention to syntactically well-formed utterances and to map such utterances into sets of underlying structures which typically contain syntactic and some semantic information. I will try to keep the treatment as independent as possible from the details of the formalism and the grammar. In cases where details are needed, I will use examples from LFG [Kaplan and Bresnan1982] or from DCG [Pereira and Warren1980]. I will assume that the structures constructed by the grammar contain an implicit or explicit representation of the underlying predicate/argument structure, i.e. they define what entities are involved in what kind of actions.

In this framework, ambiguity refers to all kinds of situations where a single utterance has more than one possible analysis. The exact set of phenomena that are covered by this generic definition depends of course on the details of the grammar. We can write a grammar that maps words into the concepts of a semantic ontology. In this case, there are ambiguous analyses for many words, and the task of identifying the right concept for each word is much harder than in situations where the set of predicates is largely determined by the words that are involved. This thesis does not try to analyze in detail the various forms of ambiguity and to device a specific disambiguation methodology for each of them. It is rather aimed at the definition of a general scheme, which can be applied on various forms of ambiguity, according to demand.

In the first part, which consists of Chapters 2 and 3, I sketch ways to *avoid* disambiguation by representing several possible interpretations in

a compact form. The work described here is based on ideas that have been developed mainly by Ron Kaplan, Martin Kay, and John Maxwell in connection with implementations of LFG. Similar approaches have been proposed in connection with logic-based NLP. However, it is useful to present the various methods for packed representations of solution spaces in a somewhat unified way, in order to highlight some underlying common principles.

In the second part, I will present a method for the partial resolution of ambiguity with the help of stochastic models. The goal of disambiguation, as seen here, is not primarily to extract a single solution out of a set of possible analyses, but to assign to every analysis a score in the form of a probability. This assessment leads straightforwardly to the selection of a "best" solution, should such a selection be required. A probabilistic assessment, however, is much more powerful. It defines a ranking, i.e. an enumeration of solutions, sorted by probability. From such a ranked list, we can find out if there is a big difference between the best and the second best solution, which may be useful to decide if the best solution should be taken unconditionally, or if further (maybe more expensive) mechanisms should be employed to reach a decision. From a probability distribution over solutions, we can also extract probabilities of certain properties of a solution, which can be useful for the collection of additional statistical training material, or to optimize the efficiency of interactive disambiguation.

A further advantage of this probabilistic scoring becomes apparent if we want to use the analysis as an input to further processing steps, such as transfer into another language, or taking some action in a dialog system. The following component may not be able to accept every input that can be produced by the syntactic analysis, or it may prefer certain inputs over others. In both situations, the existence of a stochastically ranked collection of possibilities is a good way to combine soft or hard constraints from both components to find the best overall solution.

The integration of grammatical and statistical approaches to language is an area of ongoing, intensive research activity, which did not yet lead to conclusive results. I try to offer some building blocks that may contribute to a solution, but I have to leave for the future the full integration into an NLP system and the comparison with other techniques that have been proposed in the meanwhile very extensive

literature.

Whereas the task of disambiguation between competing analyses is the main motivation of this work, there are strong connections with work on stochastic language modeling, i.e. approaches that define probability distributions on possible utterances. On one hand, I use some techniques that have been developed for language modeling and generalize them to be useful for probability distributions on syntactic dependency structures. On the other hand, the models defined here are implicitly also stochastic models for the strings these structures come with. The hope is that by exploiting syntactic dependencies, we obtain innovative and superior language models that could, at least in principle, help to increase the accuracy of speech recognition, OCR, or any other application where a distorted or ambiguous representation of natural language has to be decoded.

1.3 Structure of this Thesis

The structure of the thesis is as follows.

In Chapter 2, I review a couple of related proposals to the representation of disjunctive information in constraint-based grammars. Chapter 3 investigates the question how ambiguous results of context-free parsing may be represented compactly, and how the schemes for representing disjunctive feature information can be integrated into representations of parse forests.

In Chapter 4, I introduce some concepts and notations from probability and information theory that seem relevant for linguistic applications. I also shortly discuss ways to measure the quality of language models, and try to motivate the use of cross entropy as an evaluation criterion.

In Chapter 5, I sketch ways to estimate the probability of words, ignoring the context. Whereas this does not seem very interesting or important per se, it gives some insights into the more general estimation problem. I also investigate how morphological information can be used to obtain refined probability estimates, and how lexical data and training data from different domains can be exploited. Chapter 6 addresses simple solutions to the problem of estimating bi-lexical probabilities,

i.e. the probabilities that a pair of words appears in a given syntactic relation. I first concentrate on estimation methods that use mixtures of context-aware and marginal distributions, I discuss some of their properties, and compare them to the more sophisticated application of the Good/Turing formula to this problem given by Church/Gale. I describe how the technique of merging specific and general evidence from Chapter 5 can be further generalized and applied to bigram frequencies, leading to a novel estimation method. The chapter also investigates the question how distributional similarities between words can be exploited to get better estimates for bigram probabilities. I give an overview of the large set of techniques that have been used for this problem so far, ranging from hard over soft clustering up to nearest-neighbor-type approaches. I discuss some ways of defining suitable similarity measures, and sketch solutions to problems with time and space complexity of these approaches.

Chapter 7 investigates ways of combining the evidence from several contextual clues without exponential increase of the parameters to be estimated, which is not possible in the context-free approaches. Stochastic models can be made more context-aware by basing them directly on a description in a more expressive language, such as as constraint-based grammar formalism. In such approaches, it is in principle possible to incorporate arbitrary structural properties into the scoring function. However, this general approach makes the parameter estimation more difficult. Chapter 8 describes an experiment in which most of the techniques introduced so far are taken together to perform syntactic disambiguation on a set of parse forests produced with a large-scale LFG grammar. Finally, Chapter 9 gives a short summary of the thesis and sketches possible directions of further work.

Chapter 2

Unifying Disjunctive Feature Descriptions

2.1 Introduction

This chapter introduces the language of *feature terms* containing sorts, variables, negation and *named disjunction* for the specification of feature structures. We show that the possibility to label disjunctions with names has major advantages both for the *use* of feature logic in computational linguistics and its *implementation*. We give an open world semantics for feature terms, where the denotation of a term is determined in dependence on the *disjunctive context*, i.e. the choices taken for the disjunctions. We define *context-unique feature descriptions*, a relational, constraint-based representation language and give a normalization procedure that allows to test consistency of feature terms. This procedure does not only avoid expansion to disjunctive normal form but maintains also structure sharing between information contained in different disjuncts as much as possible. Context-unique feature descriptions can be easily implemented in environments that support ordinary unification (such as PROLOG). This chapter is based on a joint paper with Jochen Dörre and has been published as [Dörre and Eisele1990].

2.1.1 Unification-Based Formalisms

For about a decade, many formal theories of natural language have tried to describe their subject in terms of so called feature structures, i.e. potentially nested bundels of features that are assigned to words and phrases. These structures are sometimes seen as abstract linguistic objects, which are described using a suitable description language, sometimes they are given a concrete shape in form of finite automatons and regarded themselves as descriptions of the linguistic objects [Kasper and Rounds1986]. Despite such differences in interpretation, there is a consensus among the theories that linguistic descriptions should provide constraints concerning feature structures and that a set of such constraints gives a partial description of the feature structures associated with a phrase. A set of constraints defines a minimal model, i.e. a minimal structure satisfying all constraints in the set. The union of two sets of constraints not contradicting each other leads to a minimal model which is the least common extension of the models of both sets. Such minimal common extensions can be constructed by unification of the given models, hence the term unification-based formalisms.

There is also a consensus among feature-based theories that ambiguity should be described with disjunctive formulas, and most formalisms offer ways to specify them. If disjunction is present, there is usually a finite number of minimal models instead of only one. However, until now, the way such disjunctive specifications have been processed computationally was not quite satisfactory. An enumeration of the possibilities using a backtracking scheme or a chart, which corresponds to an expansion to disjunctive normal form in the underlying logic, often leads to computational inefficiency.

Approaches to improve the situation both in terms of the logic and the implementation (see e.g. [Karttunen1984, Kasper1987, Eisele1987, Maxwell and Kaplan1989]) can be subdivided in those assuming disjunctive values for features and those allowing for more general forms of disjunction. Roughly, we can state that formalisms and implementations that provide value disjunction can be implemented more easily and more efficiently, since they can exploit the fact that disjunctive information for a certain feature has no effect on other features (as long as disjunctive information does not interact with path equivalences,

see [Eisele1987]). But the restriction to value disjunction decreases the expressive power of the formalism, since disjunctions concerning different features must be stated on a higher level. Schemes providing for general disjunction allow for a more compact representation of such cases. But if disjunctive information is not local to single features, the interaction between different parts of the description is more difficult to handle (see e.g. [Kasper1987]).

The method we propose combines advantages of both approaches. It can be seen as a generalization of value disjunction, which allows for a concise description of disjunction concerning more than one feature or path. It can also be seen as an efficient implementation of general disjunction which allows to exploit the locality of disjunctive information whenever this is possible.

2.2 Feature Terms

2.2.1 Disjunction Names

The background of our approach is the simple observation that general disjunction affecting more than one feature can be reduced to value disjunction for those features, provided that the correspondence between such disjunctions can be expressed within the formalism. In order to state such correspondences, we will label disjunctions with a *disjunction name*. Take, for instance, the formula (1) that could be used to express that the directional reading of the german preposition "in" (=into) corresponds to the accusative case of the following noun phrase, whereas the static reading (=in) corresponds to the dative case. This can also be expressed by (2), where the index d_1 at the disjunction sign indicates the mutual dependence of both disjunctions. Throughout this paper, we will assume that each disjunction is labelled with a name. Even in cases where a disjunction appears only once in the initial description, naming it will help us to treat the interaction between disjunction and path equivalence correctly.

$$
\begin{array}{rcll}
s, t & \longrightarrow & A & \text{a sort} \\
 & | & x & \text{a variable} \\
 & | & \neg A \;\mid\; \neg x & \text{simple complements} \\
 & | & f{:}\, s & \text{selection} \\
 & | & s \wedge t & \text{conjunction (intersection)} \\
 & | & s \vee_d t & \text{named disjunction (union)}
\end{array}
$$

$$
\begin{array}{rcl}
[\![A]\!]_{\alpha,\kappa} & := & A^{\mathcal{I}} \\
[\![x]\!]_{\alpha,\kappa} & := & \{\alpha(x)\} \\
[\![\neg s]\!]_{\alpha,\kappa} & := & \mathcal{U} - [\![s]\!]_{\alpha,\kappa} \\
[\![f{:}\, s]\!]_{\alpha,\kappa} & := & \{a \in \mathcal{U} \mid f^{\mathcal{I}}(a) \in [\![s]\!]_{\alpha,\kappa}\} \\
[\![s \wedge t]\!]_{\alpha,\kappa} & := & [\![s]\!]_{\alpha,\kappa} \cap [\![t]\!]_{\alpha,\kappa} \\
[\![s \vee_d t]\!]_{\alpha,\kappa} & := & \begin{cases} [\![s]\!]_{\alpha,\kappa} & \text{if } \kappa(d) = l \\ [\![t]\!]_{\alpha,\kappa} & \text{if } \kappa(d) = r \end{cases}
\end{array}
$$

Figure 2.1: Syntax and Semantics of Feature Terms

$$
\begin{array}{ll}
(1) & (\text{syn} : \text{arg} : \text{case} : \text{dat} \wedge \text{sem} : \text{rel} : \text{stat\_in}) \\
 & \vee\, (\text{syn} : \text{arg} : \text{case} : \text{acc} \wedge \text{sem} : \text{rel} : \text{dir\_in}) \\
(2) & \text{syn} : \text{arg} : \text{case} : (\text{dat} \vee_{d_1} \text{acc}) \\
 & \wedge\, \text{sem} : \text{rel} : (\text{stat\_in} \vee_{d_1} \text{dir\_in})
\end{array}
$$

2.2.2 Syntax and Semantics of Feature Terms

We incorporate named disjunction into a language of so-called *feature terms* (similar to those in [Smolka1988]), where each feature term describes a set of possible feature structures. The language allows for the use of *sort symbols* $A, B, C \ldots \in \mathbf{S}$, on which some partial order $\preceq$ induces a lower semilattice (i.e. $\forall A, B \in \mathbf{S} : GLB(A, B) \in \mathbf{S}$). $\top$ and $\bot$ are the greatest and least element of $\mathbf{S}$. We also distinguish a set of *singleton sorts* ($a, b, c \ldots \in \mathbf{Sg} \subset \mathbf{S}$), which include the special sort NONE. $\bot$ is the only sort smaller than a singleton sort. The language provides a set $\mathbf{F}$ of feature symbols (written $f, g, h, \ldots$), an infinite set $\mathbf{V}$ of variables (written $x, y, z, x_1, y_1, \ldots$) to express path equivalence, and an infinite set $\mathbf{D}$ of disjunction names (written $d, d_1, d_2, \ldots$). $\mathbf{S}$,

F, **V** and **D** are pairwise disjoint. Sort symbols and variables can be negated to express negative values and path equivalence (simple negation). The restriction of negation to sort symbols and variables is not essential, since the negation of any feature term can always be reduced to these forms in linear time [Smolka1988].

Definition 1 (Feature Terms) *We define the set* **FT** *of feature terms with variables, simple negation and named disjunction by the context-free production rules given in Fig. 2.1. Letters s, t, t_1, ... will always denote feature terms.*

The semantics of our terms is defined with respect to an interpretation, which is a pair $(\mathcal{U}, \cdot^{\mathcal{I}})$ of a universe of the interpretation and an interpretation function such that:

- $\top^{\mathcal{I}} = \mathcal{U}$ and $\bot^{\mathcal{I}} = \emptyset$
- for all sorts A, B: $GLB(A, B)^{\mathcal{I}} = A^{\mathcal{I}} \cap B^{\mathcal{I}}$
- singleton sorts are mapped onto singleton sets
- for every feature f: $f^{\mathcal{I}}$ is a function $\mathcal{U} \to \mathcal{U}$.
- if a is a singleton sort and f is a feature symbol, then $f^{\mathcal{I}}$ maps $a^{\mathcal{I}}$ into $\text{NONE}^{\mathcal{I}}$

When interpreting a feature term with variables and named disjunctions, we have to make sure that the same value is assigned to each occurrence of a variable and that the same branch is chosen for each occurrence of a named disjunction. To achieve this, we introduce *variable assignments* that map variables to elements of the universe and *disjunctive contexts* that assign to each disjunction name the branch that has to be taken for this disjunction and hence specify a possible interpretation of a formula with named disjunction. Since we limit ourselves to binary disjunctions, a branch of a disjunction can be specified by one of the symbols l or r.

Definition 2 ($\mathcal{U}$-Assignment) *A $\mathcal{U}$-assignment α is an element of $\mathcal{U}^{\mathbf{V}}$, i.e. a function from* **V** *to $\mathcal{U}$.*

Definition 3 (Context) *A* context *is an element of $\{l, r\}^{\mathbf{D}}$, i.e. a function from* **D** *to the set $\{l, r\}$. The symbols κ, κ', etc. will always denote contexts.*

For a given interpretation, we define the denotation of a feature term in a context $\kappa \in \{l, r\}^{\mathbf{D}}$ under an assignment $\alpha \in \mathcal{U}^{\mathbf{V}}$ as shown in Fig. 2.1. The denotation of a feature term as such is defined by:

$$[\![s]\!] := \bigcup_{\kappa \in \{l,r\}^{\mathrm{D}}} \bigcup_{\alpha \in \mathcal{U}^{\mathrm{V}}} [\![s]\!]_{\alpha,\kappa}$$

2.3 Context-Unique Feature Descriptions

To describe the computational mechanisms needed for an implementation, we will introduce a relational language to express constraints over variables. Unlike similar approaches (e.g. [Smolka1988]), our constraint language will also be used to express disjunctive information. For this language, we will define a normal form that exhibits inconsistencies, and simplification rules that allow to normalize a given specification. Our language will provide only two kinds of constraints, one that relates a variable to some feature term (written $x \mid t$) and one that expresses that certain contexts are excluded from consideration because the information known for them is inconsistent (written $\bot[k]$).

In order to refer to sets of contexts, we define

Definition 4 (Context Descriptions) *A* context description *is a propositional formula where the constant* TRUE*, variables written $d_i{:}\, l$ and $d_i{:}\, r$ with $d_i \in \mathbf{D}$, and the operators $\wedge$, $\vee$ and $\neg$ may be employed.* $\mathbf{CD}$ *will denote the set of context descriptions. The symbols $k, k_1, \ldots$ will always denote members of* $\mathbf{CD}$*.*
The set of purely conjunctive context descriptions (not containing the operators $\vee$ and $\neg$) is denoted by $\mathbf{CD}_c$.
Each context κ satisfies the context description TRUE *(written $\kappa \models_c$ TRUE), whereas $\kappa \models_c d{:}\, b$ for $b \in \{l, r\}$ only if $\kappa(d) = b$. The meaning of context descriptions involving $\wedge$, $\vee$ and $\neg$ is defined as in propositional logic.*
If $\kappa \models_c k$, we will also say that k describes *or* covers *κ or that κ* lies in *k.*
A context description is called contradictory*, if no context satisfies it.*
Two context descriptions k, k' which are satisfied by exactly the same contexts are called equivalent *(written $k \equiv k'$).*

An important form of constraints for our approach are constraints like $x \mid x_1 \sqcup_{d_1} x_2$ which expresses that x and x_1 have to be equal in contexts where $\kappa(d_1) = l$ and so do x and x_2 in contexts where $\kappa(d_1) = r$. Such constraints are called *bifurcations* and x_1, x_2 are called (the $d_1{:}l$- and $d_1{:}r$-) *variants* of x. Assume an additional constraint $x_1 \mid x_3 \sqcup_{d_2} x_4$, then x_3 will be called the $d_1{:}l \wedge d_2{:}l$-variant of x and so on. Now, instead of accumulating constraints on the variable x which might be effective in different contexts and could interact in complicated ways, we can introduce new variables as variants of x and attach the information to them.

We will sometimes refer to a variant of a variable x without having a variable name for this variant. To this end, we will use a special notation x/k to denote the k-variant of x. Such expressions will be called *contexted variables*.

Definition 5 (Contexted Variables) *A* contexted variable *is a pair x/k where $x \in \mathbf{V}$ and $k \in \mathbf{CD}_c$.*
$\mathbf{V}_c$ will denote the union of $\mathbf{V}$ with the set of contexted variables. Elements of $\mathbf{V}_c$ will be written with capital letters $X, Y, Z, X_1, Y_1 \ldots$ To mark the distinction, we will sometimes call the members of $\mathbf{V}$ pure variables.

During the normalization of feature descriptions we will sometimes need variable substitution. If a description contains e.g. $x \mid y$, where other constraints might express conflicting information about x and y, we want to concentrate this information on one variable (say x) by substituting all occurences of y in other constraints by x. This could lead to problems when constraints attached to x and y are relevant in different contexts. One way to treat this situation correctly would be the introduction of conditional substitution (see [Eisele and Dörre1990] for details). The way we choose here is to restrict the use of variables in such a way that it is always safe to use conventional substitution.

Our trick will be to require that essentially all occurences of a variable x are relevant to the same set of contexts. We call this condition (defined more precisely below) the *context-uniqueness of variables*. We will set up the normal form and the rewrite system in such a way, that context-uniqueness of a description is maintained during the simplification process. (See [Eisele and Dörre1990] for a more detailed moti-

vation of context-uniqueness). The set of relevant contexts will be regarded as an inherent and invariant property of variables, and we will introduce a *context assignment*, i.e. a partial function $Con : \mathbf{V} \mapsto \mathbf{CD}_c$ that maps each variable in use to a purely conjunctive description of the contexts it is relevant to. We extend Con to contexted variables by defining $Con(x/k) := Con(x) \wedge k$.

In order to obtain context-unique descriptions, we generalize our feature terms so that they may also contain contexted variables.

Definition 6 (Contexted Feature Terms) *A* contexted feature term *is built according to definition 1, but where both pure and contexted variables may occur. The set of contexted feature terms will be denoted by* $\mathbf{FT}_c$. *The symbols* $s, t, t_1 \ldots$ *may henceforth also denote contexted feature terms.*
The denotation of a contexted feature term in a context $\kappa \in \{l, r\}^{\mathbf{D}}$ *under an assignment* $\alpha \in \mathcal{U}^{\mathbf{V}}$ *is defined as for usual feature terms by adding:*

$$[\![x/k]\!]_{\alpha,\kappa} := \begin{cases} \{\alpha(x)\} & \text{if } \kappa \models_c k \\ \emptyset & \text{otherwise} \end{cases}$$

We can now define the context compatibility of a feature term. This definition is somewhat technical and the reader can skip it, since our algorithm will produce only context-unique descriptions, anyway.

Definition 7 (Context compatibility) *Given a partial assignment* $Con : \mathbf{V} \mapsto \mathbf{CD}_c$, *a contexted feature term* t *is context-compatible to a context description* k *with respect to* Con, *written* $t \sim_{Con} k$, *according to the following conditions.*

$$\begin{array}{rll} A & \sim_{Con} k & \text{for arbitrary } k \in \mathbf{CD}_c \\ X & \sim_{Con} k & \text{iff } Con(X) \equiv k \\ \neg t & \sim_{Con} k & \text{iff } t \sim_{Con} k \\ f{:}t & \sim_{Con} k & \text{iff } t \sim_{Con} k \\ s \sqcap t & \sim_{Con} k & \text{iff } s \sim_{Con} k \text{ and } t \sim_{Con} k \\ s \sqcup_d t & \sim_{Con} k & \text{iff } s \sim_{Con} k \wedge d{:}l \\ & & \text{and } t \sim_{Con} k \wedge d{:}r \end{array}$$

Definition 8 (Context-unique feature descriptions) *A context-unique feature description* (x_0, CUC, Con) *is a triple such that:*

- $x_0 \in \mathbf{V}$, *called the root variable*
- *CUC is a set of context-unique constraints which either have the form*
  $\bot[k]$, *where* $k \in \mathbf{CD}$ *or*
  $X \,|\, t$, *where* $X \in \mathbf{V}_c, t \in \mathbf{FT}_c$ *and* $t \ \sim_{Con} \ Con(X)$
- *Con is a context assignment which is defined for all variables in CUC*

The semantics of context-unique feature descriptions is given by the satisfaction relation $\models_{Con}$ *between variable assignments*[1]*, contexts and constraints, which is parametrized with a context assignment.*

$$\begin{array}{lll} \alpha, \kappa \models_{Con} X \,|\, t & \textit{iff} & \kappa \not\models_c Con(X) \quad \textit{or } \alpha(X) \in [\![t]\!]_{\alpha,\kappa} \\ \alpha, \kappa \models_{Con} \bot[k] & \textit{iff} & \kappa \not\models_c k \end{array}$$

The denotation *of a context-unique f-description is:*

$$[\![(x_0, CUC, Con)]\!] := \{\alpha(x_0) \mid \alpha \in \mathcal{U}^{\mathbf{V}}, \kappa \in \{l, r\}^{\mathbf{D}} \textit{ s.t.} \\ \forall c \in CUC : \alpha, \kappa \models_{Con} c\}$$

Given a feature term t not containing the variable x_0, we can find an equivalent context-unique feature description $(x_0, \{x_0 \,|\, t'\}, Con)$ as follows. We initialize the context assignment Con so that x_0 and all variables contained in t are mapped to TRUE (they are regarded as relevant to all contexts). Then we obtain the contexted feature term t' by replacing all occurences of variables in t which are embedded in disjunctions by their appropriate variants, such that $t' \ \sim_{Con} \ \text{TRUE}$[2].

Proposition: If t does not contain the variable x_0, and if Con and t' are obtained from t as described above, then $[\![t]\!] = [\![(x_0, \{x_0 \,|\, t'\}, Con)]\!]$. For a proof see [Eisele and Dörre1990].

2.4 Normal Feature Descriptions

One way to eliminate a contexted variable (take e.g. $x/d_1 : l$) from a description is to introduce a bifurcation $(x \mid x_1 \sqcup_{d_1} x_2)$ and replace

[1] α is extended to contexted variables by: $\alpha(x/k) := \alpha(x)$

[2] In the sequel we will also assume that inaccessible disjuncts resulting from nested disjunctions with identical names (e.g. t_2 in $t_1 \sqcup_d (t_2 \sqcup_d t_3)$) are removed.

the variable by an appropriate variant (in this case x_1). Analogously, contexted variables with more complex context descriptions can be replaced by introducing several bifurcations. However, it turns out that our representation can be more compact if we allow for the use of contexted variables. But we have to prevent conflicting information from being attached to variants of a variable. Our normal form will therefore allow the use of contexted variables in certain places, but in some cases, a pure variable has to be used.

A context-unique feature description (x_0, CUC, Con) is *normal* if it satisfies the following conditions:

A) All constraints in CUC have one of the forms:

- $\bot[k]$
- $x \,|\, x_1 \sqcup_d x_2$
- $x \,|\, \neg y$, where $x \neq y$
- $x \,|\, A$ or $x \,|\, \neg A$
- $x \,|\, f{:}Y$

where $k \in \mathbf{CD}, x_1, x_2, x, y \in \mathbf{V}, Y \in \mathbf{V}_c, d \in \mathbf{D}$ and $A \in \mathbf{S} \backslash \{\top, \bot\}$

B) The following restrictions apply:

1. If $\bot[k]$ and $x \,|\, t$ are in CUC, then $Con(x) \wedge \neg k$ is not contradictory
2. if $x \,|\, A$ and $x \,|\, B$ are in CUC, then $A = B$
3. if $x \,|\, a$ and $x \,|\, t$ are in CUC, then $t = a$
4. if $x \,|\, A$ and $x \,|\, \neg B$ are in CUC, then $A \not\leq B$ and $GLB(A, B) \neq \bot$
5. if $x \,|\, \neg A$ and $x \,|\, \neg B$ are in CUC, then $A \not< B$
6. if $x \,|\, f{:}Y$ and $x \,|\, f{:}Z$ are in CUC, then $Y = Z$
7. if $\bot[k]$ and $\bot[k']$ are in CUC, then $k = k'$
8. if $x \,|\, x_1 \sqcup_d x_2$ and $x \,|\, t$ are in CUC, then $t = x_1 \sqcup_d x_2$

2.4.1 Simplification Rules for Normalization

For normalization, we have to consider all ways a context-unique feature description could fail to be normal, and we have to find an equivalent description that is in (or closer to) normal form. To this end, we give simplification rules for each possible case. Since there are many different ways to violate normal form, we get a lot of different rules, but each of them is very simple and their correctness should be easy to see.

The rules are parametrized with the root variable (which should not be substituted away) and with the context assignment, which will be extended to new variables during simplification. To facilitate notation, we use $c \;\&\; CUC$ to denote $\{c\} \cup CUC$ where CUC is supposed not to contain the constraint c, and $CUC_{x\to y}$ denotes CUC with all occurences of x replaced by y. Also, if we write $d\colon b \wedge k'$, then k' is supposed not to contain $d\colon b$. The cases we have to handle are grouped in those that treat single non-normal constraints (S) and those that treat interactions between different constraints (M).

There are S-Rules for all forms of constraints which conflict with condition A), i.e. which are of one of the forms

1. $x/k\,|\,t$
2. $x\,|\,\neg y/k$
3. $x\,|\,Y$
4. $x\,|\,t$ or $x\,|\,\neg t$, where t has the form $\top$, $\bot$ or x
5. $x\,|\,f\colon t_1$, where $t_1 \notin \mathbf{V}_c$
6. $x\,|\,t_1 \sqcap t_2$
7. $x\,|\,t_1 \sqcup_d t_2$, where $\{t_1, t_2\} \not\subset \mathbf{V}$

Among the situations in which a contexted variable x/k conflicts with normal form, we have to distinguish several cases. If $k \equiv \textsc{true}$, then the context description is irrelevant and we can replace x/k by x (Rule $S_{cu}1b$). Otherwise, if there exists already a bifurcation $x\,|\,x_l \sqcup_d x_r$, such that $k \equiv d : b \wedge k'$ for some $b \in \{l, r\}$ and $k' \in \mathbf{CD}_c$, where k' does not contain $d : b$, then we can replace x/k by the shorter term x_b/k' (Rule $S_{cu}1c$). If there is a bifurcation $x\,|\,x_l \sqcup_d x_r$ where d does not appear in k, the constraint attached to x/k is distributed over the variables x_l and x_r (Rule $S_{cu}1d$). In order to maintain context-uniqueness, the variables appearing in the constraint have to be replaced by their respective $d\colon l$- and $d\colon r$-variants. We use t/k as a shorthand for a contexted feature term, where each variable has been replaced by its k-variant, i.e. z has been replaced by z/k and z'/k' by $z'/(k' \wedge k)$ (see also rule ($M_{cu}8c$), below). Only if no bifurcation exists for x we have to introduce a new bifurcation (Rule $S_{cu}1e$). We select a disjunction name d from k such that $k \equiv d\colon b \wedge k'$ for some $b \in \{l, r\}$ and $k' \in \mathbf{CD}_c$, where k' does not contain $d : b$, we add a bifurcation $x\,|\,x_l \sqcup_d x_r$ to CUC, where x_l and x_r are new variables, and we extend Con by mapping x_l

to $Con(x) \wedge d{:}\,l$ and x_r to $Con(x) \wedge d{:}\,r$. Now we can replace x/k by x_b/k'.

The other rules handle equalities by substituting a variable by some other variable, eliminate redundant constraints, handle inconsistencies, or decompose constraints with complex feature terms into a set of simple constraints.

The cases where a pair of constraints violates some of the conditions B1–7 can be treated as for similar non-disjunctive rewrite systems (see [Smolka1988] or [Eisele and Dörre1990]). Rules $M_{cu}1-7$ handle those. When a bifurcation $x \,|\, x_1 \sqcup_d x_2$ occurs together with some other constraint on x, this could lead to a contradiction with information known about x_1 and x_2. Here, we distinguish three cases. If the other constraint happens to be a bifurcation $x \,|\, y_1 \sqcup_d y_2$ with the same disjunction name d, we get equalities between both $d{:}\,l$-variants and both $d{:}\,r$-variants (Rule $M_{cu}8a$). If the other constraint is a bifurcation $x \,|\, y_1 \sqcup_{d'} y_2$ with a different disjunction name, then the two disjunctions interact and have to be multiplied out for the variable x (Rule $M_{cu}8b$). To this end, four new variables are introduced as variants of x and new bifurcations are installed that link the new variables to those already in use. Con is extended for the new variables. In any other case, the constraint attached to x is distributed over both variants, and context descriptions for variables on the right-hand side of the constraint are introduced or adapted as required by context-uniqueness (Rule $M_{cu}8c$).

2.4.2 Soundness, Completeness and Termination

We can show that our simplification rules constitute an algorithm for the consistency (or unification) problem, which is sound and complete and guaranteed to terminate. For detailed proofs the reader is referred to [Eisele and Dörre1990]. Below, we give the key intuitions or strategies for the proofs. Soundness can be seen by inspecting the rules. Each rule rewrites a clause to one with an equivalent denotation. To show that the algorithm always finds an answer, we first observe that to every context-unique feature description that is produced during translation or normalization and that is not normal at least one of the rules applies. When the result of simplification is the single constraint $\bot[k]$ where $k \equiv \textsc{true}$, this means that the description failed to unify. In

$(S_{cu}1a)$	$x/k \,\vert\, x/k' \;\&\; CUC$	$\rightarrow_{x_0,Con}$	CUC ($k \equiv k'$ due to context-uniqueness)
$(S_{cu}1b)$	$x/k \,\vert\, t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, t \;\&\; CUC$, if $k \equiv \text{TRUE}$
$(S_{cu}1c)$	$x/k \,\vert\, t \;\&\; x \,\vert\, x_l \sqcup_d x_r \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x_b/k' \,\vert\, t \;\&\; x \,\vert\, x_l \sqcup_d x_r \;\&\; CUC$, if $k \equiv d{:}b \wedge k'$
$(S_{cu}1d)$	$x/k \,\vert\, t \;\&\; x \,\vert\, x_l \sqcup_d x_r \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x_l/k \,\vert\, t/d{:}l \;\&\; x_r/k \,\vert\, t/d{:}r \;\&\; x \,\vert\, x_l \sqcup_d x_r \;\&\; CUC$, if $(S_{cu}1c)$ does not match
$(S_{cu}1e)$	$x/k \,\vert\, t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x_b/k' \,\vert\, t \;\&\; x \,\vert\, x_l \sqcup_d x_r \;\&\; CUC$ if $(S_{cu}1a,b,c,d)$ do not match, $k \equiv d{:}b \wedge k'$, x_l, x_r are new, and $Con(x_b) := Con(x) \wedge d{:}b$
$(S_{cu}2)$	$x \,\vert\, \neg y/k \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$y/k \,\vert\, \neg x \;\&\; CUC$
$(S_{cu}3a)$	$x \,\vert\, y/k \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$y/k \,\vert\, x \;\&\; CUC$
$(S_{cu}3b)$	$x \,\vert\, y \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$CUC_{x \rightarrow y}$, if $x \neq x_0$
$(S_{cu}3c)$	$x_0 \,\vert\, y \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$CUC_{y \rightarrow x_0}$
$(S_{cu}4a)$	$x \,\vert\, t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$\bot[Con(x)] \;\&\; CUC$, if $t = \bot$, $t = \neg\top$ or $t = \neg x$
$(S_{cu}4b)$	$x \,\vert\, t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	CUC , if $t = \top$, $t = \neg\bot$ or $t = x$
$(S_{cu}5)$	$x \,\vert\, f : t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, f : y \;\&\; y \,\vert\, t \;\&\; CUC$, if $t \notin \mathbf{V}_c$ where y is new and $Con(y) := Con(x)$
$(S_{cu}6)$	$x \,\vert\, t_1 \sqcap t_2 \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, t_1 \;\&\; x \,\vert\, t_2 \;\&\; CUC$
$(S_{cu}7)$	$x \,\vert\, t_l \sqcup_d t_r \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, x_l \sqcup_d x_r \;\&\; x_l \,\vert\, t_l \;\&\; x_r \,\vert\, t_r \;\&\; CUC$ where $\{t_1, t_2\} \not\subset \mathbf{V}$, x_b are new and $Con(x_b) := Con(x) \wedge d{:}b$
$(M_{cu}1)$	$\bot[k] \;\&\; x \,\vert\, t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$\bot[k] \;\&\; CUC$, if $Con(x) \wedge \neg k$ is contradictory
$(M_{cu}2)$	$x \,\vert\, A \;\&\; x \,\vert\, B \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, GLB(A,B) \;\&\; CUC$
$(M_{cu}3a)$	$x \,\vert\, a \;\&\; x \,\vert\, \neg y \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, a \;\&\; y \,\vert\, \neg a \;\&\; CUC$
$(M_{cu}3b)$	$x \,\vert\, a \;\&\; x \,\vert\, f{:}Y \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, a \;\&\; Y \,\vert\, \text{NONE} \;\&\; CUC$
$(M_{cu}4a)$	$x \,\vert\, A \;\&\; x \,\vert\, \neg B \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$\bot[Con(x)] \;\&\; CUC$, if $A \leq B$
$(M_{cu}4b)$	$x \,\vert\, A \;\&\; x \,\vert\, \neg B \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, A \;\&\; CUC$, if $GLB(A,B) = \bot$
$(M_{cu}5)$	$x \,\vert\, \neg A \;\&\; x \,\vert\, \neg B \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, \neg B \;\&\; CUC$, if $A < B$
$(M_{cu}6)$	$x \,\vert\, f{:}Y \;\&\; x \,\vert\, f{:}Z \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, f{:}Y \;\&\; Z \,\vert\, Y \;\&\; CUC$
$(M_{cu}7)$	$\bot[k] \;\&\; \bot[k'] \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$\bot[k \vee k'] \;\&\; CUC$
$(M_{cu}8a)$	$x \,\vert\, x_1 \sqcup_d x_2 \;\&\; x \,\vert\, y_1 \sqcup_d y_2 \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, x_1 \sqcup_d x_2 \;\&\; (CUC_{y_1 \rightarrow x_1})_{y_2 \rightarrow x_2}$
$(M_{cu}8b)$	$x \,\vert\, x_1 \sqcup_{d_1} x_2 \;\&\; x \,\vert\, y \sqcup_{d_2} z \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, x_1 \sqcup_{d_1} x_2 \;\&\; x_1 \,\vert\, y_1 \sqcup_{d_2} z_1 \;\&\; x_2 \,\vert\, y_2 \sqcup_{d_2} z_2 \;\&\; y \,\vert\, y_1 \sqcup_{d_1} y_2 \;\&\; z \,\vert\, z_1 \sqcup_{d_1} z_2 \;\&\; CUC$, where $d_1 \neq d_2$ and y_1, y_2, z_1, z_2 are new
$(M_{cu}8c)$	$x \,\vert\, x_1 \sqcup_d x_2 \;\&\; x \,\vert\, t \;\&\; CUC$	$\rightarrow_{x_0,Con}$	$x \,\vert\, x_1 \sqcup_d x_2 \;\&\; x_1 \,\vert\, t/d{:}l \;\&\; x_2 \,\vert\, t/d{:}r \;\&\; CUC$ where t is not a bifurcation

Figure 2.2: Simplification Rules

any other case we can construct models from the normal form result. The basic idea is to choose a context κ which is not covered by the context description of a constraint $\bot[k]$ in our formula and 'project' the formula into this context by regarding only those constraints which are relevant to this context, thereby degenerating bifurcations to nondisjunctive bindings $x \mid y$. This nondisjunctive set of constraints can be made into a model.

In order to prove termination we construct a complexity measure for descriptions (a natural number) which is decreased in every rewrite step (see [Eisele and Dörre1990]). Here we take advantage of the fact that although there are rules which increase the number of constraints and hence seem to add to complexity, these rules also can be seen as part of an inherently irreversible process, since they distribute information attached to a variable over variables in more specific contexts. But since the number of disjunction names is limited, the contexts associated to variables can not be arbitrarily specific and hence, this process must terminate.

2.4.3 An Example

Due to lack of space, our example can not demonstrate all capabilities of the formalism, but will concentrate on the treatment of disjunction and the support of structure sharing between different disjuncts. Assume as initial feature term $f : (x \sqcap g : t_G) \sqcap h : ((x \sqcup_d y) \sqcap i : t_I)$ where t_G and t_I might be themselves complex. Translation to context-unique form will produce the description $(x_0, \{x_0|f : (x \sqcap g : t'_G) \sqcap h : ((x/d{:}\, l \sqcup_d y/d{:}\, r) \sqcap i : t'_I)\}, Con_1)$ where t'_G and t'_I might contain contexted variables if necessary. Partial normalization then produces

$$\left(x_0, \left\{ \begin{array}{ll} x_0|f : x, & x|g : x_G, \\ x_0|h : z, & z|x/d{:}\, l \sqcup_d y/d{:}\, r, \\ & z|i : x_I, \end{array} \begin{array}{l} x_G|t'_G, \\ x_I|t'_I \end{array} \right\}, Con_2 \right)$$

where the further decomposition of the constraints $x_G|t'_G,\ x_I|t'_I$ need not interest us. Since the bifurcation for z contains contexted variables, it is replaced by $z|z_l \sqcup_d z_r, z_l|x/d{:}\, l, z_r|y/d{:}\, r$, but the latter two constraints lead to the introduction of bifurcations also for x and y. Furthermore, the feature constraints on x and z are distributed over

their respective variants. We eventually get:

$$\left(x_0, \left\{ \begin{array}{lll} x_0|f:x, & z_l|g:x_G/d{:}l, & \\ x_0|h:z, & z_l|i:x_I/d{:}l, & x_G|t'_G, \\ x|z_l \sqcup_d x_r, & x_r|g:x_G/d{:}r, & x_I|t'_I \\ y|y_l \sqcup_d z_r, & z_r|i:x_I/d{:}r, & \\ z|z_l \sqcup_d z_r, & & \end{array} \right\}, Con_3 \right)$$

Although the resulting description contains contexted variables which refer to variants of x_G and x_I, we do not have to introduce bifurcations for these variables. Hence the information contained in constraints on the variables x_G and x_I is not duplicated, although both variables are used within a disjunction. However, if there would be more information on the values of the $g-$ or $i-$features of z_l, x_r, or z_r, for instance a constraint of the form $z_l|g:x'$, this would lead to the introduction of a bifurcation for x_G, and some parts of the structure embedded under x_G would have to be distributed over the variants of x_G. But the unfolding of the structure below x_G would be limited to the minimal necessary amount, since those parts of the structure that do not interact with information known about x' could make use of contexted variables.

Informally speaking, if we unify a structure with a disjunction, only those parts of the structure have to be copied that interact with the information contained in the disjunction.

2.4.4 Algorithmic Considerations

One major advantage of our treatment is its similarity with conventional rewrite systems for feature logic. Since we perform only conventional substitution of variables (opposed to conditional substitution as in [Maxwell and Kaplan1989]), our system can be easily implemented in environments providing term unification (PROLOG), or the almost linear solution of the union/find problem could be exploited (see e.g. [Aït-Kaci1984]). The only essential extension we need concerns the treatment of context descriptions. A context description contained in a contexted variable is always purely conjunctive. Hence the necessary operations (comparison with TRUE, locating, adding or deleting a simple conjunct) can each be implemented by one simple list operation. In

the constraint expressing inconsistent contexts ($\bot[k]$), k is a disjunction of the inconsistencies found so far (which themselves are purely conjunctive). This could be also represented in a list of (purely conjunctive) contexts. However, the exclusion of irrelevant constraints $x\,|\,t$, where $Con(x)$ is covered by k in $\bot[k]$, and the (final) test if $k \equiv \text{TRUE}$ involves a bit more propositional calculation. Since these tests might occur more often than the detection of a new inconsistency, it might be worthwile to use a representation that facilitates the test for entailment. In any case, the implementation can make use of fast bit-vector operations.

2.4.5 Maxwell and Kaplan's Approach

An approach which ours is especially interesting to compare with is the disjunctive constraint satisfaction procedure given in [Maxwell and Kaplan1989], because of the similar representations involved in the two approaches. They use also disjunction names and contexts to represent disjunctive constraints and propose a general transformation procedure which turns a rewrite system for non-disjunctive constraints into one which handles disjunction of constraints with the use of contexted constraints, having the implicational form $(k \rightarrow \phi)$, where ϕ is some non-disjunctive constraint. This is done by replacing every rewrite rule by its "contexted version", e.g., $\phi_1 \wedge \phi_2 \longrightarrow \phi_3$ is replaced by $(k_1 \rightarrow \phi_1) \wedge (k_2 \rightarrow \phi_2) \longrightarrow (k_1 \wedge \neg k_2 \rightarrow \phi_1) \wedge (k_2 \wedge \neg k_1 \rightarrow \phi_2) \wedge (k_1 \wedge k_2 \rightarrow \phi_3)$, where k_1 and k_2 are variables for context descriptions. There are two severe efficiency-critical problems if we want to use the outcome of this translation without further optimization. First, any rule of the generated form should only apply to a pair of contexted constraints whose contexts are compatible, i.e. $k_1 \wedge k_2$ is not contradictory. But now, since context descriptions may include conjunction and negation at any level, this test itself is an $\mathcal{NP}$-complete problem, which has to be solved before every application of a rule. The second problem concerns substitution. Consider a rule like $x \doteq y \wedge \Phi \longrightarrow \Phi_{y \rightarrow x}$. The translation produces a rule in which Φ is rewritten to both Φ and $\Phi_{y \rightarrow x}$, indexed with different context descriptions. Thus, we cannot simply perform a replacement, but instead, have to make a copy of Φ (or at least those parts of Φ

containing y). Unfortunately, this prevents also the efficient union/find method to be employed for building equivalence classes for variables instead of actual substitution. All of these problems are avoided if we let the context description of a contexted constraint depend implicitly on the variables in it through the introduction of context-unique variables. From this point of view, our method can be seen as an optimized implementation of the translated rewrite system for unification in feature logic with sorts and negation.

2.5 Conclusion

To summarize, we have presented a new unification method for the full language of feature logic including variables, sorts and negation which avoids expansion to disjunctive normal form, if possible. The basic principle is to minimize unnecessary interaction of different disjunctions by keeping them local to those attributes which they specify different values for through the introduction of disjunction names. With this treatment we avoid exponential explosion in many practical cases. A precursor of this algorithm [Dörre and Eisele1989] has been implemented and was successfully used in a grammar development environment. Besides the obvious advantage of increased efficiency, our compact representation of disjunctive information also facilitates the comparison of alternative solutions with common parts, which has been proved to be a very valuable property in our application. Our algorithm is specified in a completely formalised way as a rewrite system for which a model-theoretic semantics is given. It may seem that there are a lot of rules, but this can be explained by the following facts: we include a complete reduction from feature terms (like in Kasper/Rounds logic) to feature descriptions (as used in LFG); we handle all different types of constraints, including sorts and negation in one framework; and our rules only involve few primitive operations for which simple and fast implementations exist.

Chapter 3

Parse Forests and Disjunctive Descriptions

3.1 Introduction

In the last chapter we have introduced techniques for the representation of disjunctive information concerning features of certain linguistic entities.

It would be very useful if these techniques could give a compact representation of all possible readings of a string as a constituent of a certain category. However, this is not as straightforward as it might seem. Explicit disjunctions in grammar and lexicon are only one of several sources of ambiguity in the analysis of natural language. Another source are implicit disjunctions that show up as soon as the context-free skeleton of grammar is ambiguous and hence assigns more than one possible structure to a given string.

This chapter will investigate what role the techniques already introduced can serve in constraint-based parsing, where explicit disjunctions from grammar and lexicon and implicit disjunctions from the parsing chart interact.

Assume for a simplified example a grammar for noun compounds that has a rule like

$$\begin{array}{lll} \text{N} \rightarrow & \text{N:} & \downarrow \in (\uparrow \textsc{Compound}); \\ & \text{N:} & \uparrow = \downarrow . \end{array}$$

When used to parse a noun compound with more than two components, this rule will create multiple possible attachments. For "ink jet cartridge", we will get the correct reading "[ink jet] cartridge" as well as the wrong attachment "ink [jet cartridge]". For a longer compound like "Xerox Customer Support Center telephone number" we get 42 readings. The number of readings grows exponentially with the length of the compound[1]. The question that will be investigated here is whether and how the schemes for representing disjunctive information in a single structure discussed in the last chapter can be used to represent all the possible readings of such an expression.

Most of the work described in this chapter is taken from the literature. I will start from a very practical point of view as given in [Maxwell and Kaplan1993], in which the authors discuss various ways to organize the interaction of constraints in the context-free skeleton of the grammar with the constraints from the functional annotations of the grammar.

I will then switch to a more theoretical treatment of the problem along the lines given in [Billot and Lang1989, Lang1994]. Here, the parsing problem is seen as one of intersecting two sets of strings that are given in implicit, generative form. On one hand we have the grammar which is a finite description of an infinite set of strings (plus their syntactic structures). One the other hand, we have the input string, which is a special case of a regular language. Standard techniques from automata theory can be used to construct a grammar that generates exactly the intersection of a context-free grammar with a regular set. It turns out that the view of parsing as building this intersection has several advantages. If we remove parts of the resulting grammar that cannot generate strings, we get a concise characterization of all possible syntactic structures of the input, i.e. a compact representation of the parse forest. The size of this representation is $\mathcal{O}(n^{l+1})$, where l is the maximal length of the right-hand side of grammar rules, which means that the complexity is comparable to representations that are

[1]More precisely, the number of readings of a compound of length n is given by the Catalan number $C(n) = \frac{1}{n+1}\binom{2n}{n}$, which is of the order of $\Theta(4^n/n^{1.5})$. The approximation $C(n) \approx 0.14104 * 4^n/(n - 0.2695)^{1.5}$ has a relative error of less than 1% for $n > 1$.

constructed by standard parsing algorithms. Furthermore, the fact that the same algorithm also works for more general regular languages as input has a very important practical application, in the parsing of word lattices as they are produced by typical state-of-the-art speech recognition systems.

I will discuss some extensions of this idea for constraint-based grammars along the lines given in [Dymetman1997], where parsing is performed via local transformations of an initial program that is derived from a context-free chart into a form that syntactically excludes the possibility of unification failures. The outcome of this transformation is a compact representation of all possible parsing results.

An original contribution in this chapter is in the last section, in which I will show that the problem of disjunctive constraint satisfaction actually subsumes the parsing problem in a certain sense. I will show that it is possible to set up a parser as a pre-processor that produces for a given string w with $|w| = n$ a description $\mathcal{D}(w)$, in time and space of $\mathcal{O}(n)$, such that there is a one-to-one correspondence between the solutions of $\mathcal{D}$ and the well-formed parses of w. This result is, as far as I know, original, since the most compact representation of all possible parsing results reported in the literature need $O(n^3)$ space and time. However, it is more of theoretical than of practical importance. Even if we can represent the results compactly in a formula of size $O(n)$, the language we have to use for this representation contains disjunctive constraints, and we know that resolving such a description needs, in the worst case, exponential time. If we now show that all problems of a certain type can be rewritten in the language of disjunctive formulas over functional constraints, this is not necessarily a step towards a more efficient general treatment of these problems. It could as well just remain another proof of the expressive power of the disjunctive formalism we are using.

3.2 Phrasal and Functional Constraints

[Maxwell and Kaplan1993] investigate parsing architectures for grammar formalisms that combine of a context-free component of phrasal constraints with a separate component of attribute-value or functional

constraints. This is directly applicable to LFG, where these two descriptive levels are very explicit in a grammar, but it can also be useful for similar formalisms like Functional Unification Grammar [Kay1979] or HPSG [Pollard and Sag1987], where the phrase structure is more implicit.

The computational problems of deciding the well-formedness of a string w according to such a grammar (recognition) or of representing the possible results of syntactic analysis (parsing) can also be divided into two parts, since it must be determined whether the string satisfies the constraints of both types. For the first part, dealing with the phrasal constraints, there are well-known algorithms that work in time polynomial to the size of the input. This is essentially equivalent to the problem of recognition or parsing with a context-free grammar, which can be done in $\mathcal{O}(|w|^3)$ ([Younger1967, Earley1970, Graham, Harrison, and Ruzzo1980]). The second part of the problem, determining the consistency of a set of functional constraints, can only be solved efficiently in cases where the functional description is purely conjunctive [Nelson and Oppen1980, Knight1989]. Such cases can be solved essentially in time linear to the size of the description. The more general problem of solving a boolean combination of functional constraints is NP-complete, hence a solution in polynomial time is very unlikely to exist.

One simple strategy to solve the combined problem assumes a cascade of steps, in which first the context-free problem is solved completely and a representation of all possible parse trees is produced. From this parse forest, all the trees can be enumerated, and for each of them, the functional constraints can be solved. Unfortunately, this simple combination of well-known techniques leads to a computational disaster. The number of context-free solutions grows exponentially with the size of the input[2]. Even if each one can be tested efficiently, the overall computational complexity will be exponential.

Maxwell and Kaplan describe a set of possibilities for more finely integrated solution algorithms. The first of them, called *interleaved*

[2]Assuming that the context-free grammar or the way it is interpreted exclude the possibility of unbounded non-branching recursion. Otherwise there will be an infinite number of solutions for certain inputs.

pruning, integrates the solution of the functional constraints into the context-free parser. This has the advantage that partial analyses with inconsistent functional constraints can be determined very early and such intermediate results can be pruned from the chart immediately, which can save the computation for constructing analyses that make use of these inconsistent partial results. On the other hand, since this strategy implies to keep track of functional constraints used with the grammar rules, the intermediate representations cannot be as compact as in the context-free case. Therefore the net effect of this strategy depends on details of the grammar and the representations used, and in some cases, using interleaved pruning may actually increase the computational overhead.

They then go on to discuss in a more abstract way a set of properties of the constraint systems that can be exploited to produce different interface strategies. They identify the following properties:

Monotonicity: The fact that a inconsistent partial analysis can never take part in an overall solution. This property is an important precondition for pruning inconsistent partial results[3].

Independence: Two systems of constraints are independent if no new constraints can be deduced if the systems are conjoined. If two descriptions of partial results are independent, the overall solutions can be described without an explicit enumeration of all combinations. The compact representations of context-free parse forests make use of this property: By using a pair of pointers to the representations of adjacent constituents, we can implicitly represent all possible combinations of such internal structures without actually enumerating them explicitly. It is this property of context-free analyses that makes it possible to represent an exponential number of results in polynomial space. The methods for disjunctive constraint satisfaction as sketched in the last chapter exploit the independence of functional constraints that affect different features of some linguistic entity. A system of constraints

[3]Whereas monotonicity holds in a "pure" constraint-based grammar formalism, this is not always the case for LFG, which makes use of "meta-constraints" that can be used to require the introduction of certain descriptive elements in other parts of the description. It is the essence of these meta-constraints that they cannot be checked locally. Hence a partial analysis for which such meta-constraints are not satisfied cannot be pruned from the solution space

is said to be in free-choice form if it is a conjunction of independent disjunctions and all of the disjuncts are satisfiable. If the disjuncts themselves are in free-choice form, one speaks of nested free-choice form.

Conciseness: A constraint system (or a solution) is said to be concise if its size is a polynomial function of the input that it was derived from. Bringing a constraint system into nested free-choice form may or may not leave it concise, depending on its nature.

The authors also mention the order invariance of the constraint systems usually used for linguistic descriptions, i.e. the property that the order in which various constraints are processed does not influence the result of the processing (although the order might dramatically affect processing efficiency). Finally, they observe that the constraint systems sometimes overlap in the sense that they allow a certain freedom which one of them to use for the expression of some of the constraints. In LFG, it is sometimes possible to express a certain distinction on the level of phrasal categories or to move it instead into the space of feature values. This will not essentially change the set of possible results, but it can have a major impact on the processing efficiency.

Based on these properties, a number of new interfacing strategies are discussed. In *non-interleaved pruning*, all of the phrasal constraints are processed first, giving rise to a parse forest. This representation is then traversed in a top-down manner, which makes sure that all constituents that are encountered in this traversal actually contribute to some overall analysis of the input string. For each local subtree that is reached in the traversal, the constraints of the daughter nodes are solved first, to make sure that they actually have a nonempty set of solutions. If one of the constituents turns out not to have any solution, there is no need to traverse the remaining constituents. Only if all the constituents have found to have satisfiable constraints, their solutions are checked for mutual compatibility, which may or may not lead to a solution of the constraints for the mother node of the local subtree. In case solutions are found, these are explicitly stored with the nodes to which they belong, to avoid re-computation in case the same node is used in an alternative branch of the traversal.

Compared to interleaved pruning, the non-interleaved approach may avoid the solution of an exponential number of intermediate results that cannot contribute to an analysis spanning the complete input. On the

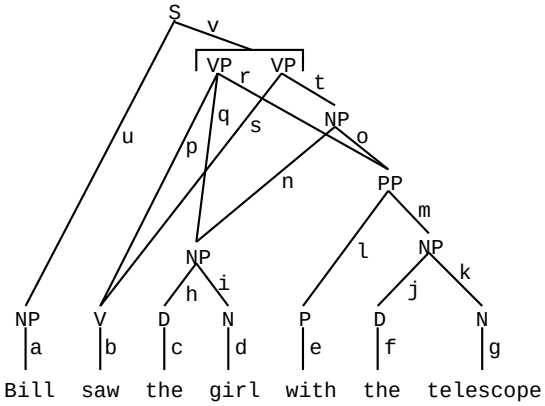

Figure 3.1: A parse forest

other hand, it may require the computation of phrasal structures that involve functionally inconsistent constituents. However, the superfluous computations can maximally require polynomial time, whereas an exponential amount of computation may be saved. So this strategy cannot lead to an asymptotic deterioration of performance, but it may actually improve efficiency.

An approach quite different from the pruning strategies is that of *factored extraction*. Here, the idea is to construct a formula that expresses the functional constraints on a solution in a concise way by factoring out common sub-formulae. For example the parse forest in Fig. 3.1, annotated with functional constraints $a \ldots v$, will lead to a representation which expands to the following formula[4]

$a \wedge u \wedge [(b \wedge p \wedge c \wedge h \wedge d \wedge i \wedge q \wedge e \wedge l \wedge f \wedge j \wedge g \wedge k \wedge m \wedge r) \vee$
$(b \wedge s \wedge c \wedge h \wedge d \wedge i \wedge n \wedge e \wedge l \wedge f \wedge j \wedge g \wedge k \wedge m \wedge o \wedge t)] \wedge v$

If common parts of this formula are recognized and factored out, the size of the formula can be reduced, and more of the constraints can be moved into the conjunctive part of the description.

$a \wedge u \wedge b \wedge c \wedge h \wedge d \wedge i \wedge e \wedge l \wedge f \wedge j \wedge g \wedge k \wedge m \wedge p \wedge [(q \wedge r) \vee (n \wedge o \wedge t)] \wedge v$

By making use of some additional knowledge concerning the equivalence of some of these constraints, the system can finally be brought

[4]The authors assume that the traversal actually leads to a re-entrant data structure which is of polynomial size, but which may be unfolded to a structure of exponential size when printed out.

into a form where the syntactic ambiguity affects only a minimal part of the description, as in the following formula.

$a \wedge u \wedge b \wedge c \wedge h \wedge d \wedge i \wedge e \wedge l \wedge f \wedge j \wedge g \wedge k \wedge m \wedge p \wedge q \wedge (r \vee o) \wedge v$

Whereas this technique works out quite nicely in the example given in the paper, it is not so clear how well it can be applied in the general case. On one hand, it would be nice to express the shared use of subformulae in an explicit way that does not resort to hidden details of the implementation, such as the use of re-entrant data structures. On the other hand, it is not clear how the specific knowledge about properties of the actual grammar that lead to the last of the simplifications shown above could be made available and exploited in the general case.

Factored pruning is a combination of factoring with the non-interleaved pruning strategy. Instead of computing all solutions for a certain constituent and storing them with that consituent in an enumeration, the common parts of the solutions are identified and stored in a factored representation.

As a last possible strategy that can be exploited to facilitate the parsing task, Maxwell and Kaplan describe *selective feature movement*, i.e. the possibility to move certain constraints from the feature space to the space of phrasal categories. Whereas expressing all necessary distinctions in the context-free part of the grammar is not possible at all or would exponentially increase the size of the grammar, it may be advantageous in selected cases where a certain distinction interacts strongly with the possible syntactic structures. They give as an example the distinction between sentences with and without preceeding complementizer. This distinction could be reflected just in the functional structure, but since it influences the way sentences can be nested, it might be advantageous to reflect it in the context-free skeleton of the grammar.

Maxwell and Kaplan experimentally evaluated a set of combinations of these strategies with several variants of unification algorithms, some of which involved contexted unification (as discussed in Chapter 2) or the method for solving a boolean combination of functional constraints that was described in [Kasper1987]. They used a corpus of 20 sentences and a grammar that was developed for independent purposes.

Their measurements show that the variant of their grammar where they have moved some constraints into the context-free skeleton was al-

ways processed faster than the original version. Non-interleaved pruning was always faster than interleaved pruning, and factored pruning was even faster. Their results concerning factored extraction were less uniform. This strategy gave the best results for the modified grammar, if used with contexted unification, but for the original grammar the results were extraordinarily bad.

A somewhat related, but little known method for representing and resolving both functional and phrasal disjunctions is given in [Block and Schmid1992]. They assume that the functional constraints can be represented using Prolog terms (with possibly shared variables) as categories. The set of partial analyses of a substring can hence be encoded in a straightforward way by an enumeration of such terms, which is related to the expansion to disjunctive normal form. As a more compact representation, they use a set forming expression, i.e. a pair $\{Template|Constraint\}$, where Template is a Prolog term and Constraint is a boolean combination of equations between Prolog terms. For instance, all possible readings of the German determiner "die" with respect to its Agreement, Gender and Case features can be encoded in the expression

```
{ det(agr(3prs,Num),Gen,Case) |
  (  Num = sg, Gen = fem
  ;  Num = pl
  ),
  (  Case = nom
  ;  Case = acc
  )
}
```

In a similar way, the notation

```
A -> X1, X2, ... , Xn | Constraint
```

is used for a grammar rule that is supposed to mean that the constituents `X1, X2, ... , Xn` can be combined to `A` if `Constraint` is satisfied. If a bottom-up parser now encounters a sequence of constituents

encoded by the terms `{T1 | C1}, {T2 | C2}, ... , {Tn | Cn}` (which are assumed not to contain shared variables), and tries to reduce them using the rule given above, the possible outcomes can be described by the expression

```
{ A  | X1 = T1, X2 = T2, ... , Xn = Tn, C1, ... , Cn, Constraint }
```

In order to check that this system of constraints is actually satisfyable, we can just "run" the constraint, which will use Prolog's backtracking mechanism to search for a solution.

A noteworthy feature of this representation is that the variables that appear in the constituents `{ Ti | Ci }` are not actually instantiated. The complex constraint built for `A` only records how the variables should be instantiated if this branch of the solution is taken. Other branches of the solution space might still refer to these constituents in a different, and actually incompatible, way. Because there is no possibility of unintended cross-talk between the solutions in a chart, they can all be collected without the need of copying partial results.

However, this representation in form of constraints will lead to increasingly complex constraints, and each processing step that needs to ensure satisfiability of some constraint might have to search over the same set of solutions of embedded constraints over and over. To avoid this type of repetition, Block and Schmid describe a method to annotate a complex constraint with a description of combinations that do not need to be searched because they are known not to contain any solutions. The key idea is that each possible way to satisfy a complex constraint can be characterized by a string over $\{l, r\}$, where the symbols indicate which branch to take for each disjunction encountered in a systematic top-down and left-to-right traversal of the formula. This characterization of a solution is called its *position*.

If solutions are generated systematically via backtracking, the positions characterizing the solutions appear in a lexicographic order. When a complex constraint is assembled and built into the chart for the first time, a search for a solution has to be made to ensure satisfiability. During this search, a certain number of disjuncts is traversed without producing an overall solution. If the set of choices made for the first

solution is kept, later uses of the complex constraint can use this set of choices as the starting point of their search for (additional) solutions. Block and Schmid use the notation `Constraint/Bound` for a constraint for which the first solution has been found with `Bound` as the associated string.

There are a number of possible refinements and variants of Block and Schmid's scheme, which are not discussed in their paper. Apparently, they assume that the satisfiability of the constraints associated to any constituent that appears is checked immediately. In that sense, they perform interleaved pruning. However, as discussed in [Maxwell and Kaplan1993], later parts of the input string may render a successfully analyzed constituent useless, and this is why in the non-interleaved pruning strategy a polynomial amount of computation with phrasal constraints can save an exponential amount of search among functional constraints.

It is possible to combine the idea of non-interleaved pruning with the Block/Schmid representation of constraints and partial solutions. During context-free parsing, a representation of all solutions according to all possible phrasal structures can be built in the way described by Block/Schmid, but no constraint is actually checked.

Once a complete phrase structure has been built, the constraint for the top-level node can be traversed and checked for satisfiability. Constituents whose constraints turn out to be satisfiable are marked by the position of the first solution, so that alternative traversals can save the time for fruitless search. Constituents whose constraints are not satisfiable at all are marked as well, so that alternative traversals can ignore this constituent immediately[5]

In this situation, a straightforward left-to-right traversal of the constituents of a local subtree does not always make optimal use of the available information, since some constituent used later might be known to be inconsistent. We can avoid traversing any left sister of such a constituent, by first checking all daughters if they have been marked as

[5]Unfortunately, the marking of constituents cannot be nicely integrated into the backtracking search for solutions via the instantiation of Prolog variables, since backtracking from a choice would then delete all the marks recorded during traversing this choice. We need a mechanism that allows to record the marks in a more persistent way, such as recording them in the Prolog database.

inconsistent, before going into the recursion[6].

There is a certain asymmetry in the way the local search space is pruned in the scheme given in Block/Schmid. When a solution for a complex constraint is found the first time, the fact that there is no other solution before it is recorded, and the associated part of the search space will never be looked at any more. However, it might be that there are many possible combinations of disjuncts in later parts of the local search space, but only very few of them (or none at all) are consistent. However, if this constituent is reused very often, this expensive search will be redone frequently.

It seems as if this overhead could be avoided by a simple extension of the basic scheme along the following lines. In the case that a solution of a local constraint cannot be used in the context in which this constraint appears, and further enumeration of possibilities is required, the positions of further solutions can be recorded in the same way as the first one. In the general case, we will hence maintain a list of solution positions, to which new positions will be added at the end in cases where the existing solutions do not suffice. Each time a solution of the constraint is needed, the list of recorded positions is scanned and each of the solutions is reconstructed and delivered. If this list is exhausted, new solutions are generated via backtracking, but the last recorded solution is used as the lower bound to start further enumeration from. Newly created solution positions are then added to the list of recorded solutions. If backtracking does not deliver any more new solutions, we record the fact that the set of solutions has been exhausted.

This method does not add to the asymptotic complexity of solving the system of constraints, but it may lead to a considerable reduction in search effort for constraint systems with many embedded disjunctions that interact so strongly that only relatively few solutions remain. One drawback is the fact that the solutions of an embedded constraint have to be produced in isolation from the context, in order to avoid that influences from contextual variable bindings prune away solutions that

[6]It is not obvious what to do in cases where long right hand sides have been split artificially into binary branching subtrees. Maybe, one would like to traverse an entire subtree to be sure that there is no inconsistent node in it before checking the consistency of any node in it. Maybe this would introduce more bookkeeping overhead than it would save computation.

would be needed in other contexts of the constraint. Abstracting from the context may lead to an increased number of solutions which then have to be filtered out. In return, we may get a collection of solutions that is reusable in other circumstances.

One could say that the approach of Block and Schmid is mainly optimized for the task of delivering the first solution as fast as possible, but might have problems with enumerating further solutions at a high rate because it might have to traverse exponential search spaces repeatedly[7]. The improvement sketched above can help to avoid this problem and can hence allow to achieve a complete enumeration of all solutions at a higher rate, without arbitrary long delays between individual solutions.

However, a more fundamental problem of this approach is the fact that we try to enumerate individual solutions at all. In situations in which many disjunctions are independent and hence an exponential amount of solutions exist, any scheme based on the individual enumeration of them will eventually lead to performance problems. The only real cure is a factored representation of the solutions in some free-choice form, from which we can tell from syntactical conditions that any way of selecting a branch of a disjunction is consistent with any possible choice in other disjunctions.

3.3 Parsing as Grammar Intersection

In this chapter, we have so far employed a rather practical view on parsing with constraint grammars. We were looking for efficient ways of finding individual solutions, but we were not very much concerned with the question how a factored representation of the solutions space should look like. Some of the technical means we have mentioned, like the use of re-entrant data structures, or the way the satisfiability of constraints is checked, have been rather close to the implementation, and we have to ask ourselves if a more formal and explicit way of dealing with these issues would not be a better choice.

[7]There is however no guarantee for a fast way to the first solution. The problem of a "trap" in the search space can as well appear before the first overall solution is found.

This is in contrast to the last chapter, where we have used a representation that allowed to encode an exponential amount of solutions in polynomial time and space, and where we have shown how to manipulate a description of the solution space in form of a set of constraints in such a way that it finally obeys certain syntactic conditions that guarantee satisfiability.

Actually, we can see both the representation of the parsing chart and the constraints attached to it in a more abstract and at the same time more uniform way, as that of a system of *definitions* that can be used to produce enumerations of individual solutions, but that could as well be seen as a computational representation of the entire solutions space that can be manipulated as a whole.

When we write a constraint of the form `Var = val1 ; Var = val2`, this can be seen as a definition for the value of Var, which has two different branches. Unless there are other constraints on the same variable that might conflict with any of these values, there is no need to actually replace `Var`, we can just leave the definition as is and wait that further information comes up (maybe in a later processing step) for which this decisions plays a role. If we have a set of variables with disjunctive definitions for each of them, we can as well leave them as is. They provide a concise description of an exponential number of possibilities.

The cases where the choices come from the ambiguity of a context-free parse are more interesting, because in such cases the constraints on individual constituents interact. Can we still represent all possible solutions to a parse in free-choice form?

For the context free case, a representation of all possible parses in a shared-packed forest can be seen as a context-free *grammar* that is specialized to the given input string, see [Billot and Lang1989, van Noord1995]. Parsing a string w according to a grammar G can, in this framework, be seen as constructing a specific variant G' of the grammar, such that $L(G') = L(G) \cap \{w\}$. More generally, we may be given a set of strings in form of a finite-state representation A and may be interested in the question whether any of the strings in $L(A)$ can be generated by G[8]. In this case, we want to obtain G' as a compact

[8]The similar question, whether G generates all the strings in $L(A)$, is undecidable in the general case

representation of all syntax trees compatible with both G and A.

A systematic way to construct G' is given in [Bar-Hillel, Perles, and Shamir1961] (cited after [Lang1994]). The construction essentially annotates the non-terminals in G with a pair of indices, which are the states in A. A grammar rule of the form $X \rightarrow YZ$ is replaced by rules of the form $\langle a_i, X, a_j \rangle \rightarrow \langle a_i, Y, a_k \rangle \langle a_k, Z, a_j \rangle$, where a_i, a_j, a_k range over all states in the finite state description A. A rule of the form $X \rightarrow x$ is replaced by the rules $\langle a_i, X, a_j \rangle - > \epsilon$ if the finite state machine A, being in state a_i can produce the terminal x and go to state a_j. Rules with longer right-hand sides can be treated similarly.

This construction needs space and time $\mathcal{O}(n^{p+1})$, where n is the number of states of A and p is the length of the longest rule right-hand-side in G. Hence this is cubic in the case where the grammar is in binary form. This construction will lead to a lot of useless and unreachable symbols, which can be pruned from the grammar in a separate step. More recent optimized parsing algorithms that produce packed representations such as [Earley1970] can be seen as instances of this scheme in which the construction of G' is interleaved with the pruning, to make sure that the nonterminals and rules introduced in G' satisfy certain conditions on their usefulness, such as actually generating a nonempty language or being accessible via a valid left-most derivation.

A nice property of this construction is that we can easily generalize it to the case of a constraint-based grammar (see [van Noord1995]). This means that we can generate a finite representation of all possible solutions in polynomial time and space. The problem with this simple approach is, however, that this representation is not in free-choice form, i.e. it is not obvious which branches in a disjunction will lead to an overall consistent solution. In general, the recognition problem for a DCG is undecidable [Pereira and Warren1983]. One way to guarantee decidability is to require the grammar to be off-line parsable, i.e. to require that the context-free skeleton for any given string contains only finitely many analyses[9]

[9] [van Noord1995] shows that off-line parsability does not guarantee decidability in the generalized case in which the input is a cyclic finite-state machine that generates an infinite language. In such cases, the recursion in A can drive the recursion in G in such a way that an indefinite number of rule applications can be be invoked,

After applying this construction to specialize a DGC G to a grammar G' that generates only strings that agree with the given input A, we can see G' as a representation of all possible analyses of strings in $L(A)$. This representation is compact in the sense that it is polynomial in the size of the input, although it contains many superfluous rules. However, this representation is not useful unless we can actually extract the solutions efficiently. The usual way to make the notion of efficient extraction more precise is to require that any choice we make in a top-down traversal of the grammar will lead to an overall solution. To ensure this property, we have to exclude the situation that a certain choice in the traversal of the representation is incompatible with choices in other parts of the chart.

In the literature there have been several proposals that use program transformations to bring a logic program into a form that guarantees the free choice among disjuncts in each disjunction. This idea has been described in [Hasida1986a] under the name "conditioned unification". The key observation is that in a conjunction of Prolog goals $G_1, \ldots, G_n$ in which each variable appears exactly once, each goal can be solved independently from the others. In such cases, the conjunction of these goals can be seen as a compact representation of a set of solutions which might otherwise be exponential in size.

A conjunction of such independent goal is called *modular*. In cases where modularity is violated due to the use of shared variables, there is a potential conflict between variable bindings in the solutions to the goals, and free choice of solutions cannot be guaranteed any more.

In this case, the goals can be partially expanded, and a new definition can be created which uses only modular goals. The partial expansion of goals is closely related to the *unfold* transformation described in [Tamaki and Sato1984, Kanamori and Horiuchi1987].

[Tsuda1993] describes the implementation of a logic programming language CU-Prolog, which uses this mechanism, together with the possibility to collapse a conjunction back onto a single goal, called the *fold* transformation, as a basic mechanism instead of SLD resolu-

despite off-line parsability. Since the DGC can be defined to do arbitrary manipulations on the arguments during the recursion, the system as a whole gets Turing power. The proof in [van Noord1995] uses a reduction from Post's Correspondence Problem.

tion. This language has been used for natural language processing in [Tsuda, Hasida, and Sirai1989, Hasida1986b]. [Nakano1991] describes a variant called *constraint projection* that allows to keep the definitions more compact and avoids the propagation of irrelevant arguments.

Independent of this, [Dymetman1997] has described a similar scheme to bring a system of definitions that describes all possible solutions of a parse with a constraint-based grammar into a form that guarantees the free choice of the alternatives. His scheme is very directly based on the Bar-Hillel/Perles/Shamir construction of a complete parse forest and on the notion of grammar transformation to turn a grammar-based description of a set of possible solutions into a normal form that excludes the possibility of inconsistent constraints for features.

[Dymetman1997] motivates his approach as a more abstract and formal specification of the algorithms described in [Maxwell and Kaplan1996b], which underlies the implementation of the parser in the XLE system [Maxwell and Kaplan1996a].

The algorithms for parsing and constraint resolution in [Maxwell and Kaplan1996b] are described in a manner that is closer to the actual implementation, making use of *disjunctive lazy copy links* to avoid copying of disjunctive constraint on feature values except for the cases where these constraints interact with information from other parts of the parse tree.

But actually, the differences between [Dymetman1997] and [Maxwell and Kaplan1996b] are far from being superficial. In the latter, the parsing result is not given in free-choice form, but in form of a contextualized feature description, similar to the descriptions in Chapter 2, where individual constraints on feature values are annotated with boolean formulas over a set of propositional variables. Certain assignments to these propositional variables may be known as inconsistent, and the information on such inconsistencies is stored in a special representation, called the nogood database.

The possibility of storing information on the interaction between feature values separately can have the advantage that the representation of feature information can be kept more compact. On the other hand, the overall representation is more sophisticated, which means that processing modules that take the parsing results as input need to be prepared to deal with this extra complexity.

The question which representation is better suited for a certain application can not be answered in a general way, as the effects that show up during the processing of larger structures depend strongly on properties of the grammars in use and on the interaction of these grammars with properties of the implementation.

So it has to remain an open question whether a more general approach based on program transformation along the lines in [Nakano1991] or [Dymetman1997] or a more sophisticated solution as the one given in [Maxwell and Kaplan1996b] will give more workable results in the long run.

3.4 "Parsing" in Linear Time and Space

In this section, I will show how a parser can construct a representation of all parsing results in time linear to the length of the string. This implies that the size of the representation has to be linear as well. The word parsing has to be put in inverted commas, since is not directly possible to extract parsing results from this representation. It is not even obvious to see if a representation of this kind contains any solutions at all. The main point of this exercise is to show that a language that allows to express disjunctions and equations between terms is powerful enough to express the parsing problem for constraint-based languages.

The material in this section is based on a combination of ideas from two sources. One of them are the considerations and papers discussed in this chapter so far. Another source has been a paper that considered resource limitations of a parser to find psycho-linguistically plausible models of parsing, [Abney and Jonson1991].

This paper points out the observation that a psycho-linguistically plausible parser cannot make active use of an unbounded amount of storage capacity, since the amount of storage available to the language processing machinery in the human brain is assumed to be finite. Actually, humans typically have problems understanding deeply embedded syntactic structures, especially if they make use of center-embedding. However, similar restrictions on the depth of left- or right-branching structures do not seem to exist. This observation has lead the authors to the conclusion that a plausible model of human parsing should be

able to process, with finite amount of memory, both left- and right-embedded recursive structures to an arbitrary depth, but that it may be allowed to fail with deeply center-embedded constructions.

A consequence of a finite amount of memory is that the processor cannot keep an unlimited amount of decisions open for review in later processing steps. Hence the amount of information that has to be extracted from a string during the parsing process has to be, in a certain sense, distributed uniformly over the string.

Many well-known parsers do not have this property. A recursive descent parser for general context free languages, that include the possibility of left-branching constructions of unlimited depth will have to be able to do an unlimited amount of recursion steps without consuming any words, which will lead to an unlimited amount of material on its stack of expected constituents[10].

A shift-reduce parser, on the other hand, has no difficulty with left-recursive structures, which it can handle with a stack of constant size. However, in order to process arbitrary deeply nested right-branching structures, it has to collect all pieces used in such a structure on the stack, until the last one appears, and only then it can start to combine them into larger constituents. Hence it needs a stack of unbounded size in order to process arbitrary deeply nested right-branching structures.

A left-corner parser can be seen as dealing with a stack of partially completed trees that are still missing a right [sic!] corner. We can imagine that a stack of such partial trees is kept, and that after reading a word, the parser may decide to use it to fill the right corner of the topmost tree on the stack (turning it into a complete tree), or push the word onto the stack separately. In any case, the top element on the stack will be complete. In the next step this complete tree may optionally be used to fill the open slot of the topmost partial tree, and this is exactly the idea on which the standard left-corner parser is built. However, if we go this way, we would also have to allow that after

[10]It is rather easy to see that this processing model is not very plausible. In order to be able to do an unlimited number of expansion steps, the parser has to be able to get into some cycle of possible expansions, but it also has to leave this cycle spontaneously at the right moment. In general, it cannot see enough of the context to make an informed decision, it just has to guess right! For this reason top-down parsers cannot be used in connection with left-recursive grammars.

reading one word, an unlimited number of such completion operations could take place, which would require an unlimited stack, even for unlimited right-branching structures.

It turns out that a simple modification of the left-corner parser can avoid this problem[11]. If we assume that the partial trees contained on the stack will never be combined directly, but there will always be some intermediate constituent, we can achieve a rather uniform distribution of the parsing actions over the string.

For each partial tree that is introduced, the parser has to decide immediately if it will be a right daughter (in this case the mother is known and the link is made immediately), or a left daughter of some node to come later. The fact that this parser has to make certain decisions earlier than a standard LC parser is not necessarily an advantage since it might have to guess in cases where the original LC parser could wait and see. On the other hand, it gives the parsing algorithm a high regularity. Essentially, for each word the following decisions have to be made:

- Choose a lexical category for the next word
- Choose if the lexical category should hang left (in this case it is unified with the missing right corner of the topmost incomplete tree) or right (in this case it is pushed on the stack as is).

In each case the stack now contains a complete tree on top, and incomplete trees below. To continue, the parser has to do two further actions:

- Chose a rule (i.e. a right sibling and a mother) for the topmost tree, turning it into an incomplete one.
- Decide if the topmost (incomplete) tree combines with the incomplete tree below it (in this case merge them) or if it will eventually be the left daughter of a node to come later (in this case leave everything as is).

[11]The following discussion and algorithm assumes the grammar to be given in Chomsky normal form.

The following little Prolog program may make this description somewhat clearer.

```
% The ``parser''

parse(String,Result) :-
        parse(String,'$EMPTY_STACK$',Result).

parse([Word|Words],Stack,Result) :-
        lex(Word,Cat),
        maybe_merge(Stack+Cat,Stack1+Top),
        ( Words=[],
          Stack1='$EMPTY_STACK$',
          Result=Top
        ; rule(M,Top,Right),
          maybe_merge(Stack1+M,Stack2),
          parse(Words, Stack2-Right,Result)
        ).

maybe_merge(Stack-Const+Const,Stack).
maybe_merge(Stack,Stack).

% example lexicon and grammar

lex(der,a(der)).
lex(frau,n(frau)).
lex(auf,p(auf)).

rule(np(np(A,N)),a(A),n(N)).
rule(np(np(N,P)),np(N),pp(P)).
rule(pp(pp(P,N)),p(P),np(N)).
```

It is straightforward to rewrite this program in such a way that the calls to the auxiliary predicate `maybe_merge/2` are not executed directly during parsing, but are instead collected in a list. This list will grow linearly with the length of the input, and in a certain, very implicit, way represents all possible parses of the input string under the given grammar. However, in order to find out if this representation

contains any solution at all, an exponential amount of search might have to take place.

When I defined this method of extracting a formula of size $\mathcal{O}(|w|)$ from a given parsing problem, I hoped that this representation would allow the application of methods of disjunctive constraint satisfaction like those described in the last chapter or the heuristic of *residuation*, as described in [Smolka1993], where constraints that can be satisfied without introducing choice points have priority over the real disjunctive constraints[12] . The idea was that in typical situations the local context around a certain parsing decision would give enough evidence to guide the parser to make certain unambiguous simplifications, so that, overall, a representation could be found in which the remaining disjunctions essentially mirror the remaining free choices, related to the real ambiguity in the utterance.

However, informal experiments with this strategy turned out that this is not the case. Typically, many disjunctions remained open, and the constraint solver needed to be forced to make decisions, in order to propagate the interdependencies through the constraint space.

3.5 Summary

In this chapter, I have given an overview on techniques that use compact representations of parse forests to represent structural ambiguities. I have reviewed some related techniques proposed in the literature that try to generalize parsing algorithms for context-free grammars to the case of constraint-based grammars, and yet preserve the polynomial computational complexity in cases where this is possible. Finally, I have given an algorithm that creates a disjunctive representation of all solutions of a parsing problem with a DCG in Chomsky normal form that used only linear time and space. However, the decision whether this compact encoding actually contains a solution remains an NP-complete problem in the general case.

[12]Residuation is also the central mechanism for treating disjunction in the CUF implementation, [Dörre and Dorna1993]

Chapter 4

Stochastic Models and their Role in NLP

This chapter gives a short and very informal motivation for the use of stochastic models in natural language processing, and hence constitutes an introduction to the second part of this thesis.

4.1 Competence versus Performance

Since the end of the fifties, initiated by Chomsky's work on formal grammars[Chomsky1957], there has been a sharp distinction between the techniques used in formal and computational linguistics on one hand, and in natural language related pattern recognition on the other hand.

Formal linguistics has mainly concentrated on questions of human language *competence*, i.e. mathematical characterizations of the totality of well-formed utterances using formal grammars and the mapping of such utterances to underlying structures. These considerations lead to the development of formalisms for the specification of grammars and lexicons and to algorithms for natural language processing, such as parsing and generation systems. However, these systems did not try to address some important aspects of human language use, such as the question which of several possible interpretations of an ambiguous utterance should be preferred in a given situation.

This omission was intentional, as it was always assumed that actual human language *performance* could be described or explained on a separate, largely independent level[Chomsky and Miller1963]. By definition, syntax does not provide enough constraints to distinguish a meaningful from a syntactically well-formed, but meaningless or implausible utterance. It is therefore obvious that many ambiguities cannot be resolved using purely syntactical knowledge sources.

The preceding two chapters assumed a rule-based description of the relation between well-formed utterances and their possible interpretations, and investigated techniques to efficiently compute and represent the set of all interpretations for a given utterance, assuming grammar and lexicon that embody a given theory of linguistic competence. These methods can be seen as the most appropriate way of dealing with ambiguity, when a model of competence is all we have.

But many applications cannot easily handle large sets of interpretations of ambiguous utterances, and it would be good to have techniques to select the right interpretation out of such sets. People who tried to build systems for speech recognition, OCR, or other applications where a distorted or ambiguous representation of natural language has to be decoded, made the observation that, in order to improve their performance, such systems also needed a way to distinguish more likely inputs from less likely ones. It seems therefore natural to complement a rule-based description of syntactic language competence with models of language performance, in order to handle such ambiguities in a useful way.

However, language performance is a much more complex field, and its description has to rely on many, and rather diverse knowledge sources, such as syntactic preferences, semantic properties of the concepts that appear in an utterance, up to pragmatic considerations involving the dialog situation and assumptions the speaker makes about the intended audience of the utterance. I think it is fair to say that only a tiny fraction of the knowledge that would be required for an adequate model has so far been captured in theory and is therefore available to computational processing. For most of the relevant knowledge sources, research has not even fixed a formal framework in which a systematic treatment could take place. Even if such a framework exists, it has been filled by content only to a quite limited amount, as the required

knowledge is often highly domain-specific, and capturing it manually in a rule-based description is an extremely difficult and expensive process.

In applications that could not be restricted to sharply limited domains, it turned out to be difficult to base such models on the deeper insights into the formal structure of language. Hence the models that were proposed for this kind of disambiguation have been extremely simplistic, based on statistics of adjacent words, ignoring essentially all the linguistic structure of an utterance. The main idea behind these models was the estimation of the probability of the next word w_i, given the context of the preceding words $w_1, ..., w_{i-1}$, and a key technique to attack the enormous sparseness problem was to ignore most of the context, except for the last two words. Since the estimation of probabilities of word trigrams from textual data still suffers from a severe sparseness problem, the main focus in the construction of these models was devoted to the question how the probability of a word w_i, given its left context w_{i-2}, w_{i-1} could be estimated from the word uni-, bi- and trigram frequencies. These models have been regarded as boring, "brute force" engineering techniques by the linguists that were interested in the structural properties of language.

Despite their structural poverty, these simple approximations were able to provide a holistic, quantitative model of some syntactic, semantic, and thematic co-occurrence patterns, as long as attention is restricted to a very narrow context. From these models, one could derive an upper bound for the entropy of the English language[Brown et al.1992], that has so far not been improved upon using more sophisticated techniques.

However, for a long time it was not clear how such a brute-force approach could be reconciled with the more ambitious theoretical questions that were investigated in formal linguistics.

Even when the ambiguity problem is a very hard one, it would be wrong to draw the conclusion that natural language processing is not feasible for the time being. Obviously, there are applications where maximal depth and correctness of the analysis are not required, or where we can get away with a (compact) representation of all possible readings of the input.

In my eyes, the best way to build up the theoretical foundation that is required for the construction of really ambitious systems that

come close to natural language understanding is by the use of suitable approximations of such theories for building concrete systems that can contribute to a systematic evaluation and specific improvement of the evolving theories.

We can see such a stochastic model as an oracle or replacement for the high-level semantic and pragmatic theories we currently do not have. We should not expect that this oracle achieves very high accuracy, but it should at least be able to separate the wheat from the chaff.

I don't think that approximations are a bad thing per se. Even if we can say very little about human language processing, we can be sure that even humans are very often not in the complete possession of all knowledge that would be required for a correct interpretation. The need to take decisions in absence of complete information is one of the basic facts of human live, and the interpretation of natural language utterances is no exception. Insofar, a combination of partial theories, combined with stochastic approximations of the missing parts might be a rather realistic model of human language perception.

The drawback that stochastic models are just approximations of the truth is compensated by the hope that their acquisition is much easier than the tedious process of a complete and detailed formalization of the knowledge that is needed.

Utterances that are ambiguous with respect to a theory of competence offer an important field of application for statistical models, where both views can be reconciled.

We can see the true interpretation of the utterance as the real signal, and the rendering of the utterance in words can be seen as the result of a transmission through some channel (perhaps a physical part of our brain?), in which some parts of the original message get lost. Seen that way, the problem of interpretation of ambiguous utterances is just another instance of a "noisy channel", in which more or less standard methods of statistical pattern recognition are applicable.

When adopting this view, the right starting point of a stochastic model is of course not the observable string of words, but some underlying structure that is seen as the original message. Since these structures can never be observed directly, but need to be postulated, based on linguistic insight, there is much freedom in the choice of the level of this interpretation and in the details of the structures that are

postulated.

I do not believe that these decisions need to be made in advance. We can as well pursue an evolutionary approach, where we start with very simple assumptions that are shared by most linguistic theories, train stochastic models on them, and only if we notice systematic errors in the decisions made by the model, we can refine the theory just as much as is needed to fix the problem. Following this general methodology, we can be sure that the details we introduce into the theory are justified by the data, and we use the data to guide the effort in the refinement of the theory.

In the late eighties and the nineties, the insight that statistical models should be used to complement purely competence-oriented formal grammars has gained wide popularity. Based on early work on formal grammars [Booth and Thompson1973], there were many papers that tried to use stochastic context free grammars for this purpose (see [Charniak1993] for a representative treatment). However, it was clear from the outset that these models could not make any use of lexical affinities between words, which, as can be seen from the ngram models, constitute a rather important knowledge source.

One of the earliest examples for the successful use of a stochastic model of such bilexical affinities has been given in [Hindle and Rooth1991]. Another important instance of the success of this idea is the work described in [Collins1996]. Popular linguistic formalisms like HPSG and LFG can be used as the symbolic skeleton of such hybrid approaches, which has been shown in [Brew1995] for HPSG and in [Johnson et al.1999] for LFG.

At the end of the nineties, even some of the most enthusiastic advocates of the theory-poor, data-driven approach have started to explore models that use somewhat richer linguistic structure[Chelba and Jelinek1998].

On can say that the historical contrast between purely statistical and purely rule-based approaches diminishes, since researchers working in both camps now recognize that a combination of both approaches is necessary for many important tasks.

Ambiguity is not the only potential use for stochastic models. Another problem that often shows up in purely rule-based accounts is that of the inherent incompleteness of the symbolic resources, due to

the difficulty of knowledge acquisition. Symbolic grammars often restrict themselves to the description of ordinary constructions and do not in detail spell out all possibilities for rare and exotic ways how people express themselves. This problem is even more striking for lexical information, where it is practically impossible to describe the inventory of words and word combination by exhaustive enumeration. So for all practical matters we have to face the situation where the symbolic resources we are using are incomplete.

Many natural language processing systems already have means to deal with this kind of situation. Part of speech taggers typically assume some probability distribution on possible categories of unknown words (where of course the open word classes are regarded as much more likely than the functional categories), and often certain orthographic aspects such as capitalization or ending influence this estimate. Even if these "guessers" alone are not very reliable, the context often provides enough additional information to make a correct classification possible.

Although this mechanism is very simple, this constitutes an example of a system that can cope with the absence of symbolic knowledge by using a quantitative model of its own imperfection. One can see this as a kind of meta-knowledge, that allows the system to keep a good balance in the relative weighting of its symbolic and stochastic knowledge sources. It is interesting to note that we can often derive from text corpora not only the raw form of lexical information, such as a word list with frequencies, but additionally also an estimate of the completeness of this resource, in form of probability estimates that the next word from the same source would already be known. This kind of meta-knowledge can be obtained by simple statistical techniques as the Good/Turing formula or cross-validation.

The remaining part of this thesis explores questions that arise when we model uncertain knowledge of various kinds with probability distributions. It is based on the assumption that probabilistic reasoning constitutes a sound formal foundation to represent quite different kinds of uncertainty in a uniform way.

Using probabilistic models, it is straightforward to represent preferences and "soft constraints", i.e. regularities that are clearly observable, but cannot be turned into clear-cut rules. But an even more important property of such models is that they can be automatically adapted to

the properties of a system from which observations can be sampled. They are able to "learn" from a set of training data, and are hence able to overcome the knowledge acquisition bottleneck. This feature makes them very useful in connection with natural language processing, as the collection of data has become rather easy, and the amounts of text that is available to the interested researcher is practically unlimited.

4.2 Stochastic Language Models

A stochastic language model is a probability distribution p over all strings in a formal language $L \subset \Sigma^*$ (over some alphabet Σ), i.e. a function p taking values in the interval $[0, 1]$ such that

$$p(w) = 0 \text{ if } w \notin L$$

and

$$\sum_{w \in L} p(w) = 1$$

In the sequel, we will sometimes use the term rather loosely, and speak also of language models when referring to a distribution over similar domains, such as a set of trees or a set of structures, generated by some suitable set of definitions.

Particularly, assume that a constraint-based grammar G defines a relation $R(G)$ between strings (over Σ) and some interpretations, taken from a set I, i.e. $R(G) \subset \Sigma^* \times I$. We will also call a probability distribution over the pairs in this relation a language model, i.e. the joint distribution p of strings and interpretations, for which

$$p(w, i) = 0 \text{ if } \langle w, i \rangle \notin R(G)$$

and

$$\sum_{\langle w,i \rangle \in R(G)} p(w, i) = 1$$

This allows us to apply stochastic language models in a situation in which some signal, consisting of a natural language utterance s, has been modified during some transmission, leading to an observation of t, and we are interested in restoring the original. If we have a stochastic

model $p(s)$ of the source signal and another model $p(t|s)$ of the distortions that happen in the "noisy channel", we can apply Bayes' decision rule and search for the input signal $\hat{s}$ that minimizes the probability of error, or maximizes the conditional probability of s, given t.

Since $p(t)$ is constant for given t, it does not matter whether we maximize the conditional probability $p(s|t)$ or the joint probability $p(s,t)$ of source and target. A simple application of the Bayes formula gives us:

$$\hat{s} = \arg\max_s p(s|t) = \arg\max_s \frac{p(s,t)}{p(t)}$$

$$= \arg\max_s p(s,t) = \arg\max_s p(s)p(t|s)$$

In many important applications of this scheme, we can assume that models of the source and models of the channel can be built independently of each other, using different types of training data from different sources. We can see this combination of two models as an example how multiple sources of uncertain information can be combined in a manner that is theoretically sound.

Language models of a simple type have already been studied in [Shannon1948], and one of the most important applications of Shannon's information theory was to quantify and maximally exploit the capacity of available transmission channels in telecommunications. Later, similar approaches have been successfully used in other applications where restoring distorted signals is important. [Duda and Hart1973] is a very good introduction into the general Bayesian approach to statistical pattern recognition, which is a generalization of the above situation, and gives applications to image analysis. The paradigm of the noisy channel constitutes the standard approach to speech recognition, starting with [Jelinek1976], and described thoroughly in [Jelinek1998]. [Kernighan, Church, and Gale1990] apply the same idea to the problem of correction of typing errors. An application of this approach to machine translation has been tried [Brown et al.1993], and is subject to ongoing research [Wu and Wong1998, Nießen et al.1998].

Stochastic language models open the way for a large number of potential applications, that are quite hard to imagine without the probabilistic component. The following enumeration can only show a small fragment.

Identification and correction of (typing) errors: The state of current technology does not allow for a reliable automatic distinction between correct and faulty utterances, due to restriction in lexical resources, grammars, and other knowledge sources. It is questionable if such a distinction will ever be feasible. However, stochastic approximations allow to make an gradual distinction into utterances that a correct or faulty with high probability. If a part of an utterance looks defective, one can use stochastic models of well-formed utterances and stochastic error models to look for the most plausible correction.

Morphological Disambiguation: Large lexicons and models of productive morphological mechanisms often lead to an abundance of unexpected morphological analyses for a given word. A stochastic evaluation can mark many of these possibilities as relatively implausible, without totally excluding them.

Syntactical disambiguation: Syntactical analysis with grammars of a reasonable coverage often lead to a large set of alternative syntactical structures, which cannot be disambiguated alone with syntactic criteria. Stochastic models of sub-categorization and lexical semantics can help to evaluate the alternative readings.

Semantic disambiguation, anaphora resolution: A deeper semantic analysis that includes a mapping of words into semantic concepts or the resolution of anaphoric references has to cope with multiple sources of ambiguities. Typically, "hard rules" for disambiguation are not available or not sufficient. Stochastic models can rank and structure the set of possible analyses, so that a correct analysis can be selected with some probability.

Machine translation (MT): As one of the most ambitious goals of natural language processing, MT has to cope with all of the difficulties already mentioned. Selection of a fitting translation usually cannot be described with "hard" symbolic conditions. Hence, stochastic models may play an extremely important role for this kind of application.

Information retrieval and text categorization: Techniques for retrieval of relevant documents from a huge amount of available texts are quickly gaining practical importance. Typically, the relevance of a document cannot be assessed by a yes/no decision. It is not astonishing that stochastic models play an important role in this area since a long time. Also for text categorization, stochastic models are essential.

Writing aids: If we assume for a moment the existence of a stochastic model that is very good in the prediction of the continuation of a partial utterance or text, it is conceivable that such a model could contribute substantially to systems that try to help people to express themselves.

The choice of a probabilistic model has several important advantages. One crucial point is the fact that one can rely on some well-established methods for the automatic acquisition of models. In the simplest case, one can use maximum-likelihood estimates of parameter values based on observed frequencies. In cases where a part of the necessary information is missing, one can use the expectation-maximization (EM) algorithm [Dempster, Laird, and Rubin1977] to replace the missing part of the data by some estimate. This has been applied for the training of Hidden Markov Models and of stochastic grammars [Baum1971, Baker1979, Pereira and Schabes1992]. Newer generalizations based on Markov Random Fields have dropped the assumption that the data is produced by a hierarchical stochastic process. Instead, characteristic properties of the data are modeled by features that can interact in rich ways. The influence of the features on the probabilities of the configurations is determined via weights. These weights are adapted to the training data by methods that iteratively optimize the weights of all features[Darroch and Ratcliff1972, Winkler1995].

Another important point is the fact that they can be evaluated in a systematic way. The notion of cross entropy provides a uniform scale to evaluate different models according to their ability to predict testing data. Here, the model with the best prediction of real data will have the smallest empirical cross entropy.

4.3 Some Problems of Stochastic Models

Even if probability theory offers a suitable formal framework to model uncertain knowledge, there is quite a number of aspects that hamper the application of standard techniques to natural language processing.

A characteristic problem in the representation of lexical knowledge are the large domains that appear. To give an example, the newspaper texts collected in the “huge german corpus” at IMS Stuttgart, containing about 200 million tokens, contain about 3 million different word

types. Although many of these forms turn out not to be words in a narrow sense, even a very restrictive selection or lemmatization cannot reduce this number substantially. At least in languages that allow to build compounds relatively freely, such as German, one should be prepared to deal with an unlimited lexical inventory.

Another problem is the fact that any useful definition of a linguistic probability (e.g. for the choice of the next word in a partially completed sentence) has to refer to the context of the utterance, which may consist of some prefix of it, but which may also include an arbitrary amount of additional information, such as topic or dialog context, in order to achieve good accuracy. Since such context also contains lexical information, the problem mentioned above will be exponentially increased, since the context is described by a large number of features, each of which has values taken from large domain.

The general problem is to determine the conditional probability $p(e|C)$ of an event e in the context C, where C can be decomposed into a sequence of features $c_1, \ldots, c_n$ such that e and each of the c_i has values from a large domain. The probability should be derived from examples, the training data, which in the ideal case should be frequent and which should be used to automatically derive the relevant parameters of the model. Due to the multitude of possible events and contexts we practically always face the situation that the number of parameters of some model is huge in relation to the available training data. Collecting larger training samples does not really help against this sparseness problem, since the domains will grow with the samples.

A possible way out of the problem is not to model the probability of words or lemmata, but to restrict the models to suitable abstractions. For a stochastic model of syntax, one could for instance abstract from the contents of the words and only take the syntactical categories into account. Another example is to map the words into a conceptual or ontological hierarchy, together with the attempt to use occurrences of such concepts as context features or as events to predict. However, this leads to a couple of difficulties.

- The abstractions are usually not given naturally, but result from a specific theoretical approach. Hence models based on such mappings are harder to compare with each other.

- The mapping of lexemes into more abstract units requires the expensive construction of large lexicons, which might be not generally useful if the underlying classification depends on specific theoretical assumptions.

- The mapping is usually not unique. Hence, available data has to be manually annotated before it can be used fro training or testing of he models.

Closely related with these problems is the difficulty to integrate existing theories and knowledge sources into a stochastic model. This makes models and training methods much more complex and requires a considerable overhead for the integration.

On the other hand, models that contain little or no theoretic concepts also face a set of specific difficulties. Such models are typically built on large amounts of training data, and in the task of predicting more data of the same source they often achieve a quality that is hard to improve upon[1]. One explanation of this is that such models realize a holistic representation of all properties of the training data that are relevant for the prediction task, that is so fine-tuned that it cannot easily be improved upon, even if the direction of of improvement is theoretically justified. The mixture contains a certain amount of local syntactical knowledge, knowledge about word frequencies in general and about local, semantically conditioned dependencies in lexical choice. This knowledge is represented in an implicit form, multiplied out into a huge number of isolated numerical values.

However, in practice the situation in which data from a specific source can be modeled using an abundant amount of training material in sufficient quality is not at all typical, for reasons already indicated above. It is much more common that one needs a model of a much more specific source, for which training data is very hard to get. In such a situation it would be good to be able to use small amount of application-specific data to adapt a large, general model optimally to the needs of a domain.

[1]To my knowledge, there exists so far no improvement on the values given in [Brown et al.1992] as an estimate for the upper bound for the entropy of English, which are essentially based on a well-smoothed trigram model.

It is easy to conceive that a stochastic model could consist of various modules that model different aspects of the data, such as genre, style, theme, etc. If such a model has to be adapted to a source that is different in certain, but not all aspects, one could reuse a part of these modules and only re-train those modules that are no more useful for the new application. The knowledge within each of the modules should be represented more compactly, using less parameters, and this would give some hope that training of single modules would need significantly less training data.

Obviously, it would be quite hard to exploit the idea of cross-fertilization between theory and applications mentioned above, if the application is based on representations that do not contain any of the existing theoretical concepts.

Finally, one of the major problems of the "theory-poor" models is the fact that today even the best existing models are very far apart from the goal we want to achieve, which is a language model that is aware of long distance dependencies and other non-trivial information from context. And we cannot expect much improvement over the the current state of the art unless we find good ways to combine linguistically informed approaches with statistical models.[2]

4.4 Statistical Models as Glue

Nowadays, as lexical resources that even cover some semantic information become more and more easily available[Miller et al.1990], one might question whether crude approximations offered by stochastic models are still needed. I see a couple of related reasons why I think that stochastic knowledge sources will be even more important in the future than they are now.

Specialized applications need to deal with highly domain-specific terminologies. The lexical information cannot be found even in large general-purpose resources. Applications get more demanding and require more sophisticated knowledge sources, that do not yet exist with

[2]Work in Jelinek's new group[Chelba and Jelinek1998] has produced a language model using a head-lexicalized stochastic grammar and reported substantial improvement in perplexity over a trigram model.

a broad coverage. More sophistication of a knowledge source means higher costs for its construction, but typically also implies more domain dependency and lower chance of re-usability.

We may want to combine several symbolic knowledge sources, with complementary imperfections, in the hope that some combination of them is more suitable for a given task than each of the components. For instance, we may have a small, domain-specific terminological lexicon with many omissions in the general vocabulary, and want to combine this with a large, general purpose lexicon, which does not contain the domain-specific terms. Still, these resources may overlap, and may occasionally offer conflicting information for their common entries. To achieve a good integration of these resources, we may want to use more or less sophisticated schemes for weighing the resources and we may still need some fall-back mechanism for dealing with cases where both of them fail to deliver the required information.

In the extreme case, we may have a large set of resources, differing in the type, size, and quality, and want to make the best use of this information for all kinds of decisions in a given task. Since the best way of exploiting such a set of resources also depends on the application domain and the task, we may need to optimize this mixture in different ways for every new application.

This new area, the integration of heterogeneous knowledge sources, will be a very important application for stochastic models for many years to come. Unfortunately, this thesis only offers some small first steps in this direction. I will show how probability estimates for words in specific domains can be obtained by an suitable combination of counts taken from large, general and from small, specific corpora. This combination can be optimized by a generalization of techniques for word probabilities, such as the Good/Turing formula and absolute discounting. I assume that this technique can be further generalized to many of the more complex situations sketched above, but a proof of this claim will require more work.

Chapter 5

Stochastic Models of Large Vocabularies

5.1 Motivation

Large corpora of naturally occuring text show that increasing the number of running words (tokens) will inevitably also increase the number of different word forms (types). Agglutinating languages or languages with compounding have vocabularies that are in principle unlimited. Even for languages with a shorter average wordlength like English it is – due to technical terms, loanwords, neologisms etc. – practically impossible to fix the overall number of word types without setting arbitrary limits.

Trying to define a probability for the occurrence of certain words leads to a number of interesting questions.

- Can one derive a meaningful probability from an observed frequency? How?
- What is the probability of observing a "new" word?
- What kinds of information on words can help to improve the estimates?
- Can estimates based on training data of a certain sort be reused for a different domain?

Some motivation for linguistic probabilities in general has been given above. The specific question discussed here is that of estimating probabilities ignoring the context, so-called "unigram"-probabilities. This problem is not very interesting when taken verbatim, since complete ignorance of the context would exclude knowledge about the domain and even knowledge of the language from which the next word will come. So let us assume we know both language and kind of text and, if we need a specific example, let us take German newspaper text. Still, the problem of estimating the probability of a word in isolation looks somewhat artificial, but one motivation for investigating this rather closely is that the techniques we will use for this problem will be generalized to the more interesting cases in subsequent chapters.

In newspapaer text, we often find very long words, and one can easily imagine a situation where a language processing system has to deal with an input that, due to some typo or transmission error, contains an unknown character. Take as an example *"Wohnumfeldverbesserungsmaßn§hmen"*, where § stands for the unknown character. We can now ask, which one of the possible completions of this word has the highest probability, and take this information to correct the error. In a case like this, the word itself provides enough information to decide this question. On the other hand, the word is very long and specific, so that we cannot expect to find this word in a text corpus.[1] Even if we find instances of possible completions, these cannot be directly translated into useful probabilities, as this example shows. (For more examples of errors in corpora, see section 5.6.) The answer we get depends on the type of knowledge we bring to bear.

5.2 The Laws by Zipf and Mandelbrot

The fact that natural languages possess an apparently unlimited supply of rare words has been known since a long time, and there have been many quantitative descriptions of this phenomenon. If words are sorted by corpus frequency and the frequency is plotted against

[1] Actually, the CD "10 Jahre TAZ" contains one occurrence of "Wohnumfeldverbesserungsmaßnehmen" (probably a typo) and "Wohnumfeldverbesserungsmaßnahmen" (measures to improve the residential area).

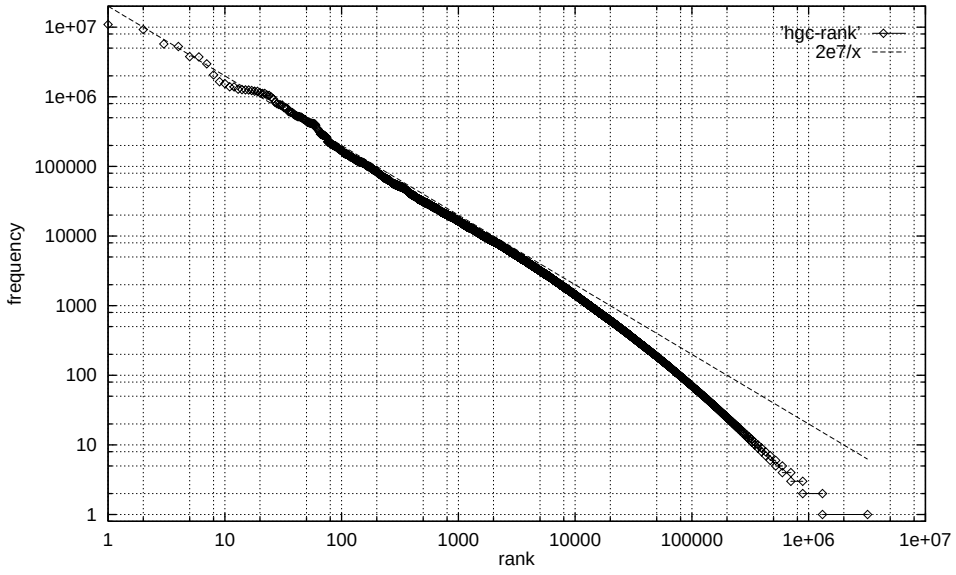

Figure 5.1: Rank-Frequency Relation in the HGC

the rank of the word in a doubly logarithmic scale, a very regular pattern shows up, which is related to "Zipf's law". According to [Chomsky and Miller1963], similar laws date back to [Estoup1916] and [Condon1928]. Zipf's law has the form

$$f(w_r) = k/r$$

where r is the rank of word w_r in a list sorted by the frequency $f(w_r)$. k is a constant that grows with the size of the sample. This formula is not very plausible for small values of r; the claim that the most frequent word is exactly twice as frequent as the next one can be easily refuted. However, for higher ranks, there usually exist considerable segments of the list where this law holds with surprising accuracy. An example is given in Fig. 5.1 that shows the rank-frequency relation for German words in the HGC[2] in a doubly logarithmic plot. For comparison, a function inversely proportional to the rank is also given.

In this diagram we can make a couple of interesting observations. For ranks below 100, the graph is somewhat unsteady, but gets rather

[2]HGC = "Huge German Corpus", a collection of several newspaper corpora that exist at IMS Stuttgart. Overall size is about 200 million word tokens.

smooth after that. Between rank 2 and about 10000, Zipf's law seems to hold approximately. Beyond rank 10000, the curve bends off, i.e. rare words appear less frequently as they are predicted by Zipf's law. For higher ranks, the function is so smooth that it seems plausible to look for a simple characterization. For very small frequencies, the function gets stair-shaped, since we can only observe integer frequencies. The last step of this stair represents the words that have been observed once (singletons).

A simple reflection shows that Zipf's law cannot hold for an infinite vocabulary. Otherwise, the probability that word w_r of rank r occurs would be $p(w_r) = k/r$. However, the harmonic series does not converge, and it is not possible to find a probability distribution that follows this law assymptotically.

However, for arbitrary $b > 1$, we have

$$\sum_{i=1}^{\infty} i^{-b} = \zeta(b)$$

where ζ stands for Riemann's Zeta function. In principle, we can generalize Zipf's law to the form $p(w_r) = \zeta(b)/i^{-r}$, which gives a distribution in which the expected frequencies diminish a bit faster than in the original form of Zipf's law. Distributions of this class have been investigated in [Mandelbrot1954]. Fig 5.2 shows each an example of the Zipfian and the Mandelbrot distribution.

However, analysis of large corpora like the HGC or the British National Corpus shows that the frequency decreases faster for higher than for medium ranks. It is not possible to obtain such a behaviour by varying the parameters of the above formula.

Zipf and other authors tried to explain the regularity of the frequency distribution as a consequence of deeper reasons, such as the principle of least effort in human behaviour and the like. However, as has already been shown in [Chomsky and Miller1963], we do not need to resort to such complex explanations, since similar characteristics can be observed in data sampled from very simple stochastic processes. Take as an example an unlimited sequence of independent and uniformly distributed binary digits and group it into pairs, such that the pairs `00`, `01`, `10` and `11` each appear with the probability 0.25. Regard

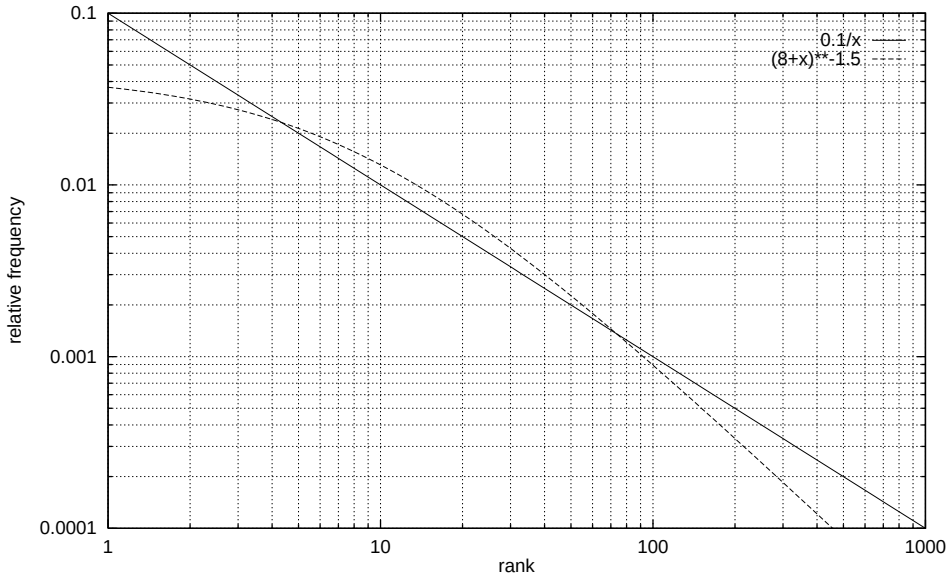

Figure 5.2: Zipfian versus Mandelbrot Distribution

00 as a word boundary, and treat all symbols between subsequent word boundaries as a word. This leads to a denumerable infinite vocabulary and to a probability distribution on it that we can very easily study analytically. For each $n \geq 0$, we have 3^n different words of length $2n$, each of them with the probability $0.25^{(n+1)}$. The words of length $2n$ occupy the ranks $(3^n + 1)/2$ up to $(3^{n+1} - 1)/2$.

Fig. 5.3 shows word frequencies in a random sample of size 10^6 according to this distribution (points), the values that would be theoretically expected (steps), as well as the plot of the function $2.5 * 10^5/x^{log(4)/log(3)}$ that connects the centers of the steps. Ignoring the zigzag shape, this distribution follows the Zipf/Mandelbrot law with $b = 1.2618595072$.

The plot shows that the expected steps show up very nicely in the experimental results for higher frequencies. For medium frequencies, the random variation leads to a smoothing of the steps. For even higher ranks, the fact that we can only observe integer frequencies leads to another sequence of steps, which is a kind of "discretization noise". It is interesting that the observed frequencies deviate systematically from

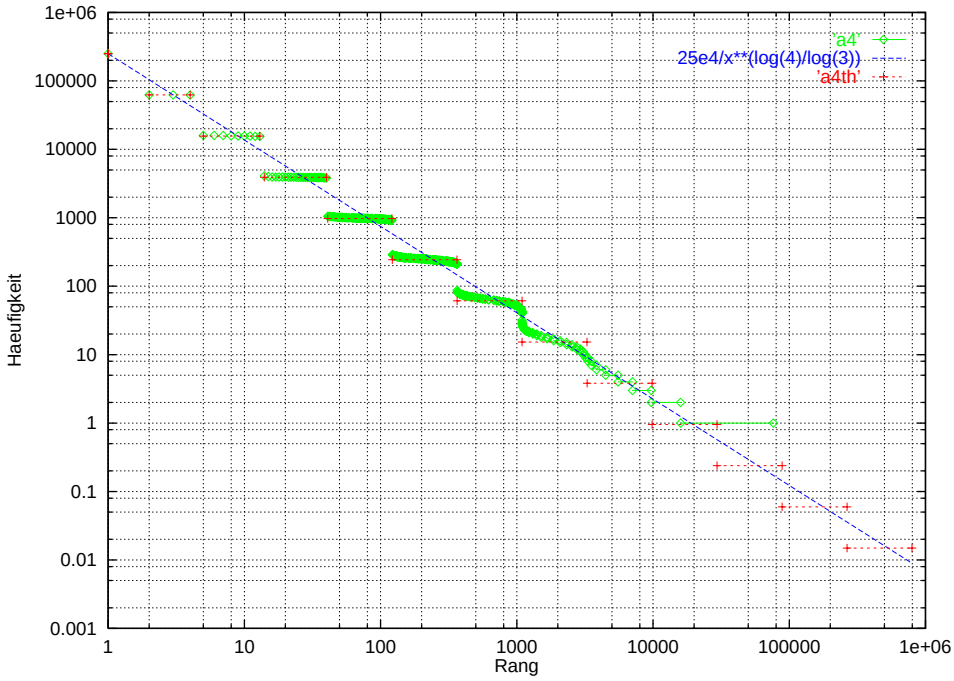

Figure 5.3: Rank-Frequency Relation of a Random Text.

the expected frequencies in such a way that the last step is shifted rightwards from the expected position. Apparently, many values have been observed exactly once, even if their expected frequency is considerably lower than one. The reverse case, that a word is less frequent than expected also exists, but not that often.

For determining the probability of rare events, it would be very helpful to know the course of the underlying distribution in the domain of low frequencies. But just in this domain the observed data is subject to high uncertainty, and needs considerable correction.

5.3 The Good/Turing Formula

A situation where we know details of a probability distribution, as described in the last section, does not often occur in practice.

In any practical situation, we only have a set of examples, the training data, and we are trying to predict properties of the underlying distribution from this observation. Of course, all stochastic models we can come up with are bound to be a quite drastic idealization of the real facts; a simple mathematical model that is built in the hope that it might predict some observable properties of reality. No claim whatsoever is made that the real processes underlying human language production or perception can be described in such simple-minded ways.

This chapter is based on a quite drastic idealization. We model the generation of natural language utterances as a multinomial distribution on a countably infinite vocabulary, i.e. a memoryless stochastic process that emits random words according to this distribution. The only question we are interested in is the estimation of a word probability, given the frequency counts of some observed sample.

Of course, the assumption of a multinomial distribution does not constrain the possibilities very much. Given the observations, any distribution that assigns a non-zero probability to the observed events could have produced the data. How can we choose any of them to predict the probabilities of additional observations from the same source?

For each distribution ϕ, we can calculate the probability that the

model produced the data as

$$P(f_i|\phi) = \binom{\sum f_i}{f_i} \prod_i \phi_i^{f_i}$$

and we can ask ourselves for what model this probability would be maximal. However, we are varying the model, and not the possible outcome of a fixed model, so this method does not define a probability distribution over possible models. Usually, this function on models is called the *likelihood* of the model, given the data. We can try to look for the model that maximizes the likelihood, and call this the *maximum likelihood estimate (MLE)*. It turns out that in our case the maximum likelihood estimate is well-defined and is exactly the one that assigns each event its observed relative frequency in the training data.

The MLE seem to have the advantage of being independent of additional assumptions. However, in our application, the MLE has the very unfortunate property to predict that any event that was not seen in the training data has probability 0, i.e. is impossible. It is obviously not very useful under circumstances in which we have to assess the probability of unseen events.

To get more useful probability estimates, we have to address three questions.

- What is a realistic probability that a future event will be *new*, i.e. of an event type that is not contained in the training sample.
- How should we reduce the probability estimates of the observed events, in order to save the necessary probability mass.
- How should we distribute this probability over the (usually infinite) set of possible, but unseen events.

For the first two of these problems, we can apply a method that has been originally suggested by A. Turing and futher investigated and published by I.J. Good [Good1953]. Here, the set of all possible events is partitioned into equivalence classes according to the observed frequencies, and an identical probability estimate is assigned to all members of a class.

Let n_f be the number of types that have been observed exactly f times in a sample of size $N = \sum_f f * n_f$. For each observed frequency f we want to obtain a corrected value f^*, as an expected frequency of an event seen f times. The Good/Turing formula states

$$f^* = (f+1) * n_{f+1}/n_f$$

[Church and Gale1991] gives a proof that this formula is correct up to a relative error of $1/N$ if it uses the expected values of the class sizes instead of the actually observed values.

In our application, there is an infinite number of unseen event types, so the formula is not directly applicable for them. However, we can still take n_1/N as a (typically very good) estimate of the probability that the next event is new.

The formula can also be motivated by considering the following experiment. Assume that the sample is partitioned randomly into a training set and a cross-validation set. An estimator should be evaluated by its ability to predict the cross-validation item from the training set. To avoid random effects of the selection of the cross-validation set and to exploit the available data optimally, we take each possible choice of the cross-validation set of size 1 into account and try to find values for f^* that give optimal prediction in all these cases. More precisely, we are looking for a function $\cdot^*$ with the constraint $\sum_f f^* * n_f = N$ that maximizes the leaving-one-out probability

$$\prod_f \left(\frac{(f-1)^*}{N-1} \right)^f$$

Intuitively, each event seen f times in the overall training data has to be produced f times in this leaving-one-out scenario, but each time, this has to happen based on the evidence given in the reduced training set, where this event was seen only $f-1$ times. Hence the class of events seen $f-1$ times has to produce $f * n_f$ events, and should get an according fraction of the overall probability, which leads directly to the formula given above. Effectively, this formula redistributes all counts obtained for the class of events seen f times onto the class of events seen $f-1$ times.

It is a very nice property of this formula that it does not make any assumption on the underlying distribution. We can, for instance, take a sample of size $k * n$ from a uniform multinomial distribution with n possible event types, where we expect k events for every type. The actually observed frequency of any type can then be approximated by a Poisson distribution with parameter k, i.e. we expect $\frac{k^f}{e^k f!}$ event types that appear exactly f times. If we would actually observe this class sizes, and put them back into the Good/Turing formula, we would get out the correct value k, no matter the value of f we actually observed[3].

In order to see what the Good/Turing formula does, we can apply it to word counts from the HGC as shown in the following table.

[3]The independence of the Good/Turing formula from the actual distribution does not seem to be widely known, and one can sometimes see people derive one of them from the other. [Samuelsson1996] investigates the somewhat artificial question what kind of distributions would be invariant under the correction by the Good/Turing formula, and explores similarities of this class and zipfian distributions.

N_f	f	f^*	
∞	0	– NA –	p_{neu}=0.00919051
1882337	1	0.44760	
421266	2	1.32056	
185435	3	2.36247	
109521	4	3.33639	
73081	5	4.41825	
53815	6	5.32905	
40969	7	6.49057	
33239	8	7.29092	
26927	9	8.41089	
22648	10	9.33698	
19224	11	10.3695	
16612	12	11.4302	
14606	13	12.4367	
12975	14	13.5468	
11718	15	14.1744	
10381	16	15.5589	
9501	17	16.6890	
8809	18	16.9984	
7881	19	18.2160	
7178	20	20.1721	
6895	21	19.7793	
6199	22	22.2579	
5999	23	22.1877	
5546	24	23.1563	
5137	25	24.3753	
4816	26	24.5164	
4373	27	27.4109	
4281	28	26.7713	
3952	29	28.3451	
3734	30	29.1735	
3514	31	31.9909	
3513	32	30.7361	

Applying the leaving-one-out idea avoids some problems with overfitting of the data, but it should not be seen as a panacea. One potential problem is that the training sample might not actually consist of in-

dependent observations. Assume for instance that, after some sample has been drawn, a certain fraction of it is mistakenly counted mutliple times, a problem which happens in practice more often than one would assume. This will increase the apparent size of the sample, but it will not increase the number of different word types. Most likely, it will actually decrease n_1, the number of events seen exactly once. Consequently, the estimated probability of unseen events will be much lower than for the original source[4].

Another problem with the Good/Turing formula is that we do not know the expected class sizes, but have to derive them from the actual observations. Directly using the observed sizes leads to unmotivated jumps in the estimates, and for higher frequencies, for which the class sizes are typically 0 or 1, the estimates alternate between 0 and ∞.

This problem shows up very clearly in the example given above, where the events seen 21 times have a smaller probability than the events seen 20 times, which is quite implausible.

[Good1953] makes detailed suggestions on how to smooth these sizes, but these are mathematically complex, and the theoretical status of these methods is not clear.

There are a number of intuitively plausible constraints on the shape of the discounting function, e.g. that it should be monotonically increasing and "smooth" in a certain sense, but these consideration affect the intended outcome of the Good/Turing formula, and it is not obvious how the input to the formula should be treated in order to get results with the required properties.

5.4 Generalized Absolute Discounting

[Ney and Essen1993] observe that for frequencies of words and n-grams, the application of the Good/Turing formula usually results in "absolute discounting", i.e. in a reduction of the observed counts by a constant amount that is usually somewhere between 0 and 1. Informally, this can be explained by the fact that in Zipfian distributions, there are more unlikely events which might have appeared more than expected

[4]However, the new estimate will still be realistic if we assume that the process we are modeling is sampling of the original data including the random duplication.

to produce a certain number of observations than events which might be "underrepresented". It would be interesting to investigate more closely the relation between the discounting constant on one hand and the properties of the distribution and the sample size on the other. But as Ney and his colleagues show, the leaving-one-out method is a good way to get reliable estimates in many practical situations, so that a deeper and theoretically more satisfying justification does not seem to promise immediate practical advantages.

In the sequel, I will present a variant of the absolute discounting method that has some advantages over it, insofar as it also works quite well in cases of non-Zipfian distributions and it leaves more flexibility for the probability of events seen once, which often do not fit into the general pattern.

The idea is to compose a discounting function as a linear mixture of several components, where each of them concentrates on the prediction of events from different frequency classes. We can find the optimal mixture of these components using the EM algorithm on cross validation data or under the leaving-one-out training regime.

We specify

$$f^* = \lambda_0 m_0(f) + \lambda_1 m_1(f) + \lambda_2 m_2(f) + \lambda_3 m_3(f)$$

where $\sum_i \lambda_i = 1$ and the models m_i are defined as follows:

$$m_0(f) = \begin{cases} N/n_0 & \text{if} \quad f = 0 \\ 0 & \text{if} \quad f > 0 \end{cases}$$

$$m_1(f) = \begin{cases} N/n_1 & \text{if} \quad f = 1 \\ 0 & \text{if} \quad f \neq 1 \end{cases}$$

$$m_2(f) = \begin{cases} N/(t - n_1) & \text{if} \quad f > 1 \\ 0 & \text{if} \quad f < 2 \end{cases}$$

$$m_3(f) = \begin{cases} (f-2)/(N - 2 * t + n_1) & \text{if} \quad f > 2 \\ 0 & \text{if} \quad f < 3 \end{cases}$$

Here, t stands for the number of observed event types $t = \sum_f n_f$. To make things a bit more perspicuous, we can visualize the four contributions as in Figure 5.4.

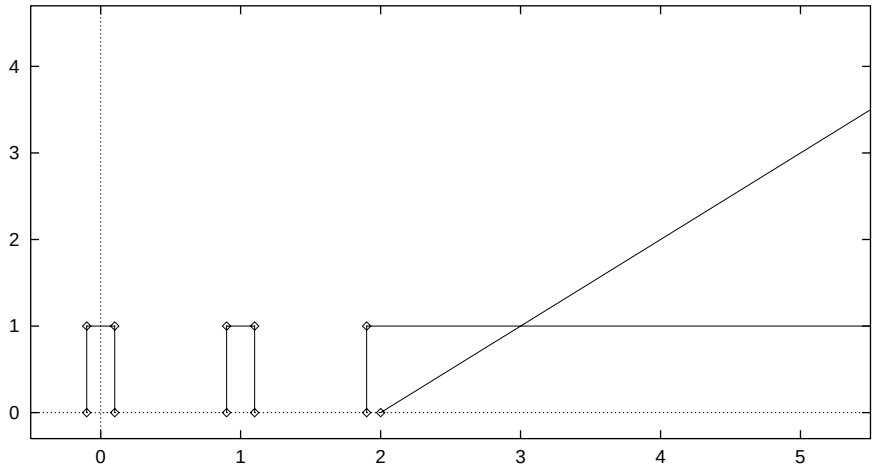

Figure 5.4: Components of the discounting mixture

Since we are looking for a mixture of four components, there are only three free parameters to optimize. Two of them can even be given in a closed-form expression, since we can read off from the data the contributions of the first and second component[5]. It is only the relative weight of m_2 and m_3 that is a bit harder to determine, since many observations (all events for event types with $f > 2$) can be generated by either of them.

However, if we make the additional plausible assumption that the slope of the resulting function should be 1 for higher frequencies, we obtain an easy way to compute all needed parameters in a simple way, without any iteration[6].

[5]The values we get for $f = 0$ and $f = 1$ are exactly those we get via direct application of the Good/Turing formula. We implicitly assume that the observed n_f for $f < 3$ are exact enough to be used as input to G/T, which is definitely the case for large enough samples from Zipf-shaped distributions, but may not hold otherwise.

[6]In section 5.7 we will discuss a more general setup where we do not assume that all the training data comes from the same source we want to characterize. This situation will lead to functions with slopes $<< 1$.
Using plausibility arguments in this context raises another question. If a slope of 1 is most plausible and leads to a nice formula, what about the situations in which the exact calculation leads to significantly different results?

5.5 Decomposing Words

We have explored various ways to estimate word probabilities by "discounting" the relative frequencies of words from the training data, but so far, we have said little about the question how the saved probability mass should be redistributed among the infinite number of words we have not seen. It is clear that we cannot treat all of them equally and we would like to exploit the training data and possibly other knowledge sources to make this redistribution more informed. One might wonder what role this question can play in connection with the problem of disambiguation, since in such settings the words are normally given. One answer is that once we have a good way of estimating probabilities for unseen words, we may try to apply it to the somewhat more difficult problem of estimating the probability of a unknown word with some given property (e.g. its part of speech). If we have good estimators for a couple of properties, and we find an unknown word in an utterance to be analyzed, we can apply Bayes rule to find a probability distribution on the properties of this word. This can be useful to guess properties of unknown words in a more flexible, example-driven way. Another answer is that we could use good probability distribution over words in a spelling-correction setting. Conventional, dictionary-based spelling correctors are often tedious to use because of the limitations of their lexicons. Having a good way of predicting new words might help. A third answer is to use such a module for the analysis of long, domain-specific compounds in languages like German. Very often, a complex technical term cannot be analysed because one of its compounts is so specific that it is not in the lexicon. The risk of such failures grows with the length of the word, with the effect that the failure rate is highest for the most informative words. Having a good way to distinguish plausible unknown components from implausible ones could help to deal more gracefully with this kind of problems.

Among the character strings that do not occur within the 3193939 types in the HGC we find quite normal words that would have considerable chances to appear in a similar corpus, but also very long words that are very unlikely or nonsensical. Most of the possible strings would not be recognized as words at all, and although they could in principle appear in a similar corpus, their probability is extremely small.

There are a several properties of such strings we could use to model the probability of an unknown word, such as wordlength, probabilities of characters and character sequences that appear in it, or knowledge about possible components and about morphology.

However, we have to keep in mind that we want to model unknown words, that appear rarely in corpora. We should not expect that a morphological component will be as useful for such a task as for the analysis of average words. We cannot fully rely on knowledge-based approaches, but always have to consider the exceptional cases which are not covered by our rules (or lexicons). We should also keep in mind that we eventually want to model distributions on words in a way that makes strong use of contextual information. Hence it does not make much sense to develop a sophisticated methodology for a simplified task if there is no way to apply it to the really interesting cases.

Therefore, I want to specify in the sequel some simple models for rare words. They are Markov models, i.e. stochastic processes based on finite automatons, that contain, for every possible state, a probability distribution on possible transitions into new states. In every state, there is additionally a certain probability to end the symbol sequence. This defines a probability distribution on the strings that can be produced, and it depends on a suitable choice of the parameters if this distribution is a good model of the real data. Of course, the parameters should not be assigned by hand, but rather be optimized based on a training sample.

We can exploit the fact that the frequency based model given above gives away a constant part of the sample frequency for the model of new words. These contributions can be used by the Markov model. The event types can also be seen as a representative sample of the data that should be modeled by the finite automaton. We can hence expect that training on the observed *types* can give better results than training on *tokens*, because the Markov model is supposed to complement, but not to replace the frequency-based estimates.

A very simple Markov model could be based on trigrams of characters, such that the contexts (i.e. prefixes of the words) are collected into equivalence classes according to the last two characters they contain, and for each such class, a probability distribution on the next character is estimated on the basis of relative frequencies. However, we get, in

a smaller scale, similar problems as for word n-gram models. On one hand, the context contains clearly too little information, and the model collapses contexts that should be kept apart. On the other hand, many two-character combinations are possible, but so rare, that a distribution on the possible continuations cannot reliably be estimated. Therefore, a suitably smoothed trigram model cannot be much more than a very superficial ad-hoc approach to the problem of missing words.

Using morphological information

The trigram model mentioned in the last chapter can give only a very rough approximation. It is very simple to compose meaningless and hence very unlikely words from frequent character trigrams, and the estimated probability of such words will be relatively high. It looks plausible that a linguistically motivated description of morphological regularities will give us a much better distinction between possible and impossible words, which will enable us to estimate the probabilities in a much better way.

I do not want to go into the fine details of morphological theories and formalisms, but instead assume a morphological description and the implementation of the assiciated tools as given, and show how a probabilistic model can be built out of these ingredients. Since the seminal work of [Koskenniemi1983] and the implementation of suitable compilers [Karttunen1994], the framework of the 2-level formalism has been used to build morphological description for a large number of languages (see http://www.xrce.xerox.com/research/mltt/Tools/morph.html for an overview). These contain stem lexicons for inflected parts-of-speech, the description of inflectional classes, and rules that describe phonological effects that modify the strings when certain sounds or characters are concatenated. These knowledge sources can be represented as finite automata. Some of these automata operate on two levels, which means that all of the transitions are annotated with pairs of symbols for the two levels. Eventually, all the knowledge sources can be compiled into a large finite state transducer (FST), that defines the mapping between orthographical strings and strings of symbols on a more abstract level of representation, the *lexical* level.

The relation between these strings is a regular relation

[Kaplan and Kay1994]. The FSTs and the mapping they define can be seen as a declarative, reversible description, that can be used in both directions, i.e. given one of the strings, the set of strings that can potentially be associated to it can be computed efficiently. The FST can also be used to enumerate the set of all pairs in this relation, provided this set is finite.

Because such an FST contains all the lexical knowledge that went into its construction, it is typically rather large[7], so that a direct manual inspection or manipulation of it is not feasible.

There are two possibilities to use a morphology component to build a stochastic language model.

One one hand, we can model the process of generating strings directly based on the given finite automaton. We can turn the FST into a Markov model by annotating the transition with probabilities. This simplifies the task, because the structure of the model is given, and all we have to add are the probabilities. Of course, this simplicity can be a disadvantage, if it turns out that the stochastic regularities cannot be described in the framework of the given structure. This is to be expected, because the tools that generate the FST will collapse states that cannot be distinguished symbolically. But the actual use of words may show regularities that cannot be expressed in this way[8]. Generally, even in morphology, complex semantic and pragmatic aspects play an important role, and we cannot expect that all relevant knowledge can be encoded in an atomic state.

Alternatively, we might use a stochastic model of the strings that appear on the lexical level, that is independent from inner structure of the FST, which then would merely provide the translation between the stochastic model and the strings that can actually be observed.

[7]The German morphology component DMOR by Anne Schiller that has been used for the experiments described in the sequel, leads to a cyclic transducer that has 139758 states and 389476 transitions.

[8]A mass noun as *"Bier" (beer)* might almost always be used in its singular form, but since the plural form also exists, the morphology may use a standard continuation class for it. There are also many verbs for which the first or second person forms are extremely rare for semantic reasons, as e.g. for *"regnen" (to rain)*. Building noun compounds should consider possible semantic relations between the components.

This architecture looks much more plausible from a linguistic point of view. The stochastic model of the lexical strings should allow to model arbitrary interrelations that exists on the lexical level, and we could abstract from the idiosyncracies of the morphology. Unfortunately, this leads to a number of problems.

We cannot assume that a consistent stochastic model of lexical strings will be transduced into a consistent stochastic word model by the given FST, since this might produce for a given lexical string a set of words with a cardinality different from 1. In this case, we would not have a probability distribution on the set of words.

A model that is composed from a lexical generator and a morphology transducer will only work for words that are known to both components. Such a system will have far more problems with coverage and robustness than a integrated model.

The interrelations between the components of a noun compound may be very important, but their treatment requires techniques that will be introduced in the Chapters 6 and 7.

For these reasons, it seems reasonable to start with a model that is formally rather simple, before trying to account more accurately for more complex effects.

A simple stochastic model of morphology can be obtained by annotating all possible transitions of an FST with probabilities. If the probabilities of all transitions that leave a certain state plus the probability that the transducer will stop in this state sum up to 1 for all states, the stochastic transducer defines a probability distribution on all pairs of strings that can be produced by the model[9].

Let $\langle w_{lex}, w_{surf} \rangle$ be a pair of lexical and surface strings that are produced by the model, we have

$$\sum_{w_{lex}, w_{surf}} p(w_{lex}, w_{surf}) = 1$$

This joint distribution of string pairs can also be used for the comparative assessment of competing analyses, by computing for a given

[9]To be quite precise, we have to require that for each state there is a path to a final state with positive probability. Otherwise, some fraction of the probability could be lost on infinite cycles in a dead end of the transducer.

surface string w_{surf} the values

$$p(w_{lex}|w_{surf}) = \frac{p(w_{lex}, w_{surf})}{\sum_{w_{lex}} p(w_{lex}, w_{surf})}$$

Since the number of possible analyses (=lexical strings) of a given word is typically small, the sum in the denominator can be computed in a simple procedure based on backtracking.

Since a morphological FST has typically a large number of states and transitions, it is essential for a practical application of this idea to acquire the parameters of the model[10] automatically from training data. Ideally, we would like to use a set of morphologically disambiguated word occurrences, i.e. a collection of pairs of the form $\langle w_{lex}, p_{surf} \rangle$. However, data of this form is not easy to obtain, because the morphological interpretation of a word occurrence is determined by the context, and can only be disambiguated manually or by the application of very powerful tools, if considerable accuracy is required. As an alternative, one can use a method that needs only surface strings and estimates probabilities for the associated lexical strings. This is a classical application of the expectation maximisation (EM) algorithm[Dempster, Laird, and Rubin1977, McLachlan and Krishnan1998] that uses a preliminary model to compute expectations for missing parts of the training data. The data completed in such a way can them be used to estimate the parameters of a new model.

By iterating these steps, the model (and the estimated part of the training data) is modified until no further increase of the likelihood is achieved. Each step in the iteration is guaranteed not to decrease the likelihood of the observed data, which leads to an overall convergence towards a (possibly local) maximum likelihood estimate. The EM algorithm is applicable under very general conditions and has frequently been used in connection with tasks related to NLP with stochastic models. Applications range from the training of Markov models in phonetics (in form of the forward-backward (FB-) algorithm [Baum1971, Juang and Rabiner1992]), of stochastic context-free grammars (inside-outside algorithm[Baker1979, Lari and Young1990]), over

[10]The number of free parameters of the model is the number of transitions minus the number of non-final states.

```
For each Transition k :
        f[k] := 0;
For each word w_surf in the training corpus:
        p_cum := 0;
        For each possible analysis A:
                p_cum := p_cum + p(A);
        For each possible analysis A:
                For each transition k used for A:
                        f[k] := f[k] + 1/p_cum;
```

Figure 5.5: Simple EM-algorithm for transition probabilities

schemes for soft classification ([Rooth1995, Saul and Pereira1997]). Here, FB and IO algorithms are specific refinements of the EM algorithm that make it possible to apply the algorithm in polynomial time to the problems for which a naive implementation would lead to exponential complexity. This can be achieved by storing and re-using intermediate results. The characteristic property of these algorithms is therefore the combination of the EM algorithm with techniques of dynamic programming[Aho and Ullman1972].

Athough the model used here is formally a Markov model, it is in this case not essential to use the FB algorithm, because for each given word the number of analyses is so small that an explicit enumeration is easily feasible. This leads to the version of the EM algorithm given in Figure 5.5. $p(A)$ is the product of the probabilities of all transitions used in A.

Training of the model can be speeded up considerably by using frequency word list and analysing only once each word type that appears in the corpus. The estimates of the transition probabilities have to be multiplied with the frequency of the word. Using this procedure, I estimated transition frequencies for the 388918 edges in the DMOR FST and used them to construct weighted transducers. I kept the counts after the first and after the third iteration of the EM algorithm. The latter gave much sharper distinctions, but there are also cases in which repeated applications of EM leads to the suppression of desired readings.

Because the probabilities cannot be more than very crude approx-

imations, for which the order of magnitude is much more important than the exact value, I used a logarithmic representation for them in 8 bits per transition. This keeps the space consumption for the weights much more compact than for the FST proper[11] Fig. 5.6 shows the morphological analyses of the word *"Träger" (Noun: carrier, Adj: lazier)*, together with the conditional probabilities assigned to the readings after the first and the third iteration of the training procedure.

It is clearly visible that the iteration reduces the probability of the adjective readings, which are rather implausible. Some of them are reduced by several orders of magnitude. However, in order to decide if the resulting values are actually realistic, one would have to manually investigate a rather large number of corpus references for a larger number of words, which is difficult to justify.

The original goal of the stochastic extension of DMOR was not to perform an absolute stochastic assessment of the joint probabilities of the pairs $p(w_{lex}, w_{surf})$ and hence implicitly estimate $P(w_{surf})$. The goal was to assess for a given word w_{surf} the set of possible readings, i.e. to construct a model for $p(w_{lex}|w_{surf})$ for those words that are known to the morphology. As a sole stochastic word model, the weighted DMOR automaton cannot be used, because of its incomplete coverage. It would have to be complemented by mechanisms that assign a positive probability to arbitrary strings. One possibility would be the mixture of the weighted DMOR FST and a stochastic model based on character trigrams, which would solve the problem with missing coverage, albeit in a very crude way. The probability of a word would then be given by

$$p(w) = \lambda p_{morph}(w) + (1 - \lambda) p_{tri}$$

where p_{morph} is the model based on morphology, p_{tri} that based on trigrams, and λ is an empirically optimized mixing factor[12].

[11] In my implementation, I used an encoding that uses 4 Bytes per transition and that doesn't need extra space per state. This is somewhat less compact than the encodings used by Xerox, but very easy to use.

[12] Under the assumption that the morphology-based model spents much less probability on non-words than the trigram model, we can estimate that the optimal mixture factor is approximately equal to the coverage of the morphology, i.e. about 95%.

$p(A\|w, 1.It.)$	$p(A\|w, 3.It.)$	A
0.145830	0.169450	Träger+NN.Masc.Akk.Pl
0.145830	0.169450	Träger+NN.Masc.Akk.Sg
0.145830	0.169450	Träger+NN.Masc.Nom.Pl
0.145830	0.169450	Träger+NN.Masc.Nom.Sg
0.145830	0.169450	Träger+NN.Masc.Dat.Sg
0.137645	0.150962	Träger+NN.Masc.Gen.Pl
0.013656	0.000156	*träg+ADJ.Comp.Adv
0.013656	0.000156	*träg+ADJ.Comp.Pred
0.013656	0.000139	träge^ADJ.Pos+NN.Fem.Dat.Sg.St
0.013656	0.000139	träge^ADJ.Pos+NN.Masc.Nom.Sg.St/Mix
0.013656	0.000139	träge^ADJ.Pos+NN.NoGend.Gen.Pl.St
0.013656	0.000139	träg^ADJ.Pos+NN.Fem.Dat.Sg.St
0.013656	0.000139	träg^ADJ.Pos+NN.Masc.Nom.Sg.St/Mix
0.013656	0.000139	träg^ADJ.Pos+NN.Fem.Gen.Sg.St
0.004301	0.000139	träge^ADJ.Pos+NN.Fem.Gen.Sg.St
0.004301	0.000139	träg^ADJ.Pos+NN.NoGend.Gen.Pl.St
0.002151	0.000074	*träge+ADJ.Pos.Masc.Nom.Sg.St/Mix
0.002151	0.000059	*träge+ADJ.Pos.Fem.Gen.Sg.St
0.002151	0.000059	*träge+ADJ.Pos.Fem.Dat.Sg.St
0.001916	0.000052	*träge+ADJ.Pos.NoGend.Gen.Pl.St
0.001279	0.000031	*träge+ADJ.Comp.Pred
0.001279	0.000031	*träge+ADJ.Comp.Adv
0.001139	0.000015	*träg+ADJ.Pos.Fem.Dat.Sg.St
0.001139	0.000015	*träg+ADJ.Pos.Fem.Gen.Sg.St
0.001075	0.000015	*träg+ADJ.Pos.NoGend.Gen.Pl.St
0.001075	0.000014	*träg+ADJ.Pos.Masc.Nom.Sg.St/Mix

Figure 5.6: Probability Estimates for Morphological Analyses

This external component leads to a complete coverage of all possible (and impossible) words, but the price for this coverage is that the fraction of theoverall probability that is distributed over unknown words is spread over all possible strings in a very crude manner.

A better model can be achieved based on the assumption that even for words that eventually cannot be analyzed, the morphology FST is able to construct reasonable partial analyses. One can observe that relatively many failures happen for long noun compounds where not all of the compounds appear in the stem lexicon[13]. If one were able to recognize and isolate such unknown parts of a word, and still analyze the rest, one could even (especially if the stem of the last component of the word is known) perform a plausible guess of part-of-speech and morpho-syntactic features, which are determined by the last component.

This leads to a model in which the missing coverage of the original transducer is rectified via the addition of transitions. It looks very promising to induce such new transitions empirically from frequency word lists, which would lead a way to a semi-automatic, data-driven extension of the morphological lexicon. However, this leads to a couple of problems, for which I did not find satisfactory solutions so far.

A transition in a morphological FST carries relatively much information (source and target states, pair of labels). If one assumes that all transitions are possible, and tries to estimate probabilities for them from training data, the number of parameters to estimate is overwhelming in relation to the training data that is available. A related problem is teh fact that the number of possible analyses of a given word will be astronomical.

A possibility to cope with these problems is the following model. The transducer is extended with one additional state, which is linked to all other states by transitions in both directions. These transitions carry an empty label on the surface side, and a special mark on the lexical side, which leaves a trace in the analysis, should such a detour be taken. This extension needs two additional parameters per original

[13]Of course, this effect is limited to languages like German that build such compounds. It is particularly visible in connection with technical or scientific text, in which compounds tend to be longer and contain domain-specific components.

state, which would approximately double the number of parameters int the DMOR example. With such an extensions, the FST can spontaneously jump from any given state to any other state, without consuming any input symbols[14]. Such a model can accept strings that are composed from characters that appear on some surface transition. The transition probabilities can be optimized via the EM algorithm. Training about 500000 parameters from a frequency word list with some million entries seems still to be feasible.

However, the approach that has been sketched above, where all possible analysis paths have been enumerated in backtracking, is not feasible any more under circumstances that would introduce massively ambiguous results for each word in the training data, because the complexity would grow exponentially with the length of the input word. Training the probabilities of such a model therefore requires the application of the forward-backward algorithm, which allows all necessary computations be performed in time that grows linear with the length of the input word.

5.6 Coping with Errors in the Training Data

Working with large amounts of naturally occurring text frequently leads to problems related to the low quality of these texts. The dilemma lies in the fact that we need, on one hand, large amounts of data to train useful models, but that this requirement makes it on the other hand necessary to use data of questionable quality. Cleaning such data manually is typically not feasible, because it would require an enormous amount of manual labour. This seems to be a considerable handicap for data-driven construction of linguistic resources. For instance, in a list of approximately 70000 types that have been annotated as verb lemmata in the HGC, more than half of the entries turned out to be

[14]Some unknown *stem* would hence have to be composed from pieces of known stems, which may lead to somewhat arbitrary results. More regular results could be achieved using transitions that originate and end in the new state, carrying all possible surface labels. On the other hand, reusing larger parts of existing words might lead to a model that adapts better to the properties of the actual data.

wrong [Eckle, personal communication]. Here, several sources of errors, like typing errors in the raw data, errors in tagging and lemmatizing act in combination. If one collects the data as is into a word list, one random error takes as much space in this list as a correct entry that is used quite frequently. In a way, correct information gets more and more diluted with random errors, that generate much more word types[15], and it requires much effort to extract high-quality information from such a collection. On the other hand one of the welcome properties of statistical models is the fact that influence from poor data is more or less randomly distributed, whereas the characteristic properties of the correct data stands out against this random noise via steadily growing frequency.

This very general remark is not very helpful in a situation where we have a large list of forms, each appearing once, and we know from

[15]The HGC contains the following spellings of one lemma, for which the morphological paradigm before the German spelling reform provided only four possible inflectional forms:

3219	Untersuchungsausschuß		
49	Untersuchungsauschuß	138	Untersuchungsausschüsse
14	Untersuchungssausschuß	2	Untersuchungsausschusse
5	Untersuchungausschuß		
2	Untersuchungsauuschuß	87	Untersuchungsausschüssen
2	Untersuchungsaussschuß		
2	Untersuchungsausschuss	804	Untersuchungsausschusses
1	untersuchungsausschuß	18	Untersuchungsauschusses
1	tersuchungssausschuß	4	Untersuchungssausschusses
1	Untesuchungsausschuß	4	Untersuchungsausschußes
1	Unteruchungsausschuß	3	Untersuchungausschusses
1	Untersuchuungsausschuß	1	ntersuchungsausschusses
1	Untersuchungsausschß	1	Untersuchungsauswschusses
1	Untersuchungsausschußhaben	1	Untersuchungsaussschusses
1	Untersuchungsausschu	1	Untersuchungsausschüsses
1	Untersuchungsausschiß	1	Untersuchungsauschußes
1	Untersuchungsauschuss	1	Untersuchunggsausschusses
1	Untersuchungsaus	1	Untersuchungaausschusses
1	UntersuchungsAusschuß	1	Unterschungsausschusses
1	Untersuchunfsausschuß	1	Unterrsuchungsausschusses
1	Untersuchtungsausschuß		
1	Untersuchsausschuß		
1	Unterssuchungsausschuß		
1	Unterschungsausschuß		

checking samples that it contains a large number of errors, but also so much useful information, that we cannot afford to ignore this data.

Basically, this is a problem of classification, and not so much one of data modeling. One way to address it is to make use of an explicit model of typical errors, such as insertions, deletions, swapping of adjacent characters or confusion with other characters [16]. Such an error model can be implemented as a weighted FST, and it can be used to represent, for a given correct form a probability distribution over possible misspellings of it, or vice versa.

If we make additional use of a stochastic model of correctly spelled words, that reserves an appropriate probability for unseen words, we could compare, for each unseen word, the probability that it is actually a new, correct word, with the probability that it is a misspelled variant of a different word. Using this technique, one could then again derive a new, improved model of correct words (and maybe also a modified wrror model), and iterate this until now new errors in the data is found.

This idea leads to the practical question how the possible corrections of a potentially misspelled word can be found efficiently. The minimal number of modifications that is necessary to transform a given word into a different one is called their Lehvenstein distance. It can be calculated using a dynamic programming algorithm, the complexity of which grows with the product of the lengths of both words. However, given the huge)or even infinite) number of forms that could serve as a potential correction of a given word, the naive enumeration of all possibilities is not feasible.

If the known words are represented in a finite atomaton or an FST, one can interpret this automaton in a way that tolerates a certain number of deviations. One can regard this a dynamic way of composing the Markov model for creating the corret word (the source model) with a Markov model of introducing errors (the channel model).

Although it is looking plausible, it might not be feasible to perform the composition of the two finite automata in a compilation step, because the resulting transducer would get too big. Therefore, the com-

[16]Depending on the source of the errors, one can use specific confusion matrices that give higher probabilities for the confusion of keys that are adjacent on a keybord, or for the confusion of phonetically similar characters etc.

position has to happen at run time, where the number of states that can be reached is limited by the given word.

Experiments with this kind of error-tolerant, FST-based access to the lexicon are described in [Oflazer1996], which is however not based on a probabilistic model. Similar methods for error correction have already been described in [Kernighan, Church, and Gale1990].

It looks plausible that a suitable stochastic model of unknown words and of typing errors can lead to a significantly better error correction. However, this requires a considerable implementation effort, and I cannot present concrete results in the framework of this thesis.

5.7 Exploiting Heterogeneous Training Data

The examples discussed in the last sections are somewhat artificial. We assumed to have large (huge) amounts of training data from a source we assume to be homogeneous[17], and we use this to estimate a probability distribution on events that are produced from exactly the same source. By lumping everything together we may ignore very useful distinctions, and this will lead to a measurable decrease in performance.

In this section, I will show how we can apply the same underlying ideas (optimizing a simple parametrized discounting function) in a slightly more complex situation in order to achieve refined estimates of measurable higher quality.

Assume the realistic situation where we have some task connected to a certain application domain (lets take newspaper articles about computers for illustration), for which we need good estimates of word probabilities[18]. Assume that we do not only have a limited amount of text material S_1 from this domain, but also a (perhaps huge) amount S_0 of unrelated text (articles from a larger newspaper collection, say), which might or might not contain relevant parts. Can we make any reasonable use of this additional knowledge source to get more reliable

[17] I.e. we choose to ignore the inner structure, although we know it exists, in order to simplify our treatment.

[18] Still ignoring the local context

probabilities?

A priory, this is hard to decide. The huge data source may actually be very heterogeneous, and there may be material from our target domain hidden inside, with no cheap way to quantify its ratio. We may also be unlucky and have no task-specific data in S_0, but having some contribution from it might still help to fill the gaps with respect to the general language that exist in S_1. So it seems that S_0 should be useful, but we don't know to what extent.

It is a nice property of statistical approaches that questions like this can sometimes be answered by a closer look at a sufficient amount of data.

We can start by a argumentation similar to the leaving-one-out scenario that motivated the Good/Turing formula. For each event type, we have two statistics f_0 and f_1, the number of occurrences in both corpora. We will lump all event types with the same statistics into an equivalence class and estimate uniform probabilities for all events of a class.

We can set up a contingency table that gives for each pair of frequencies $\langle f_0, f_1 \rangle$ the number n_{f_0,f_1} of event types that appeared with these frequencies in the two samples.

The following data uses word frequencies from two sources: a 2.1 million word sample from the German weekly magazine "Computerzeitung" and a 103 million word sample from the German part of the "European Languages News Corpus" from Linguistic Data Consortium[19].

	0	1	2	3	4	5	6	7	8	9	10	>10	f(cz)
0	infty	57793	7901	2792	1364	824	531	378	248	183	159	944	
1	646995	5200	1099	467	262	142	102	67	67	49	33	270	
2	220569	3053	715	346	188	103	74	55	28	23	20	160	
3	100518	2093	524	229	116	64	43	36	27	22	18	110	
4	60318	1570	455	170	95	64	39	34	25	22	7	77	
5	38700	1418	355	141	78	53	34	16	22	17	8	91	
6	30237	1128	338	140	77	48	27	17	11	11	12	94	
7	23192	1049	299	119	54	52	23	11	12	12	8	67	
8	18368	835	276	107	62	29	34	19	10	6	3	39	
9	15319	749	248	85	57	29	23	20	11	9	6	48	
10	12483	686	222	94	61	37	24	14	7	8	5	50	
>10	146229	16831	7831	4599	3227	2244	1742	1350	1108	928	754	11174	

f(ELNC-D)

To estimate a probability $p_1(f_0, f_1)$ that a word with statistics $\langle f_0, f_1 \rangle$ will show up in a further sample from S_1, the leaving-one-out

[19] I did not use the larger HGC, because is subsumes the CZ corpus.

argumentation will advice us to redistribute the counts that fall into the cell $\langle f_0, f_1 \rangle$ into the cell $\langle f_0, f_1 - 1 \rangle$, with other words, to assume that the expected frequency $x_1(f_0, f_1)$ of such events is

$$x_1(f_0, f_1) = \frac{(f_1 + 1)n(f_0, f_1 + 1)}{n(f_0, f_1)}$$

This method, without further smoothing of the class sizes, will lead to the following estimates:

	0	1	2	3	4	5	6	7	8	9	f(cz)
0	--------	0.273	1.060	1.954	3.021	3.867	4.983	5.249	6.641	8.689	
1	0.008	0.423	1.275	2.244	2.710	4.310	4.598	8.000	6.582	6.735	
2	0.014	0.468	1.452	2.173	2.739	4.311	5.203	4.073	7.393	8.696	
3	0.021	0.501	1.311	2.026	2.759	4.031	5.860	6.000	7.333	8.182	
4	0.026	0.580	1.121	2.235	3.368	3.656	6.103	5.882	7.920	3.182	
5	0.037	0.501	1.192	2.213	3.397	3.849	3.294	11.000	6.955	4.706	
6	0.037	0.599	1.243	2.200	3.117	3.375	4.407	5.176	9.000	10.909	
7	0.045	0.570	1.194	1.815	4.815	2.654	3.348	8.727	9.000	6.667	
8	0.045	0.661	1.163	2.318	2.339	7.034	3.912	4.211	5.400	5.000	
9	0.049	0.662	1.028	2.682	2.544	4.759	6.087	4.400	7.364	6.667	
10	0.055	0.647	1.270	2.596	3.033	3.892	4.083	4.000	10.286	6.250	
>10	0.115	0.931	1.762	2.807	3.477	4.658	5.425	6.566	7.538	8.125	

f(ELNC-D)

A subjective inspection of these estimates indicates that the frequency in the auxiliary corpus actually has some systematic influence on the estimates of word probabilities. This influence is most drastic for words that do not appear in CZ at all, here the expectation varies between 0.008 for words that appear once in ELNC-D and 0.115 for words that appear more then ten times, i.e. a factor of more than 14. For words that appear in CZ exactly once, the estimate varies between 0.273 for $f_{ELNC-D} = 0$ and 0.931 for $f_{ELNC-D} > 10$. For words that appear repeatedly in CZ, the class sizes get small and the estimates get somewhat noisy. One can say, however, that a good estimate is $f_1 - d$, where d varies between 1 for $f_{ELNC-D} = 0$ and 0.3 for $f_{ELNC-D} > 10$.

We can summarize the situation by saying that the word frequency in a large, general corpus can have a significant influence on word probability estimates, and that we can quantify this effect quite well for small frequencies, if the corpora are not quite small.

Before using such estimates for a stochastic model of the domain-specific corpus, especially if the corpora are not so large, it is advisable to smooth out the jumps that come from random fluctuations in the class sizes. We can use the techniques already discussed above and do a

smoothed estimation for each level of f_0, or we can even lump together intervals of f_0, in order to get more reliable estimates. It would be even nicer to get a parametric characterization of $x_1(f_0, f_1)$ as function of f_0 and f_1 and to optimize the parameters, but it is not quite obvious from which set of functions to choose. We may assume that we have an additive mixture of two parts, that only depend on f_0 and on f_1, respectively. But this assumption would have to level out the effect of f_0 for positive f_1 in order not to exaggerate its effect for $f_1 = 0$. A good compromise may be to model the first two columns independently, and assume a additive mixture of the remaining columns.

It should be noted that this use of cross-corpus frequencies is of course by far not the only possible application of this knowledge source. The frequency ratio between general and specific text corpora can be used as an important input for the extraction of terminological information, i.e. words that are much more likely in the domain than they are in general. This terminological application is in a certain way complementary to the use suggested here, since it has to concentrate on higher frequencies, for which this ratio can be determined reliably. Our application concentrates more on the small frequencies, and the information we extract should be rather seen as a statistical property of the corpora involved than as information on the single words involved.

5.8 Summary

In this chapter, I have presented various techniques that can be used to model stochastic models of large vocabularies. I first investigated typical properties of distributions on such large domains, and gave some motivation and some examples of the use of the Good/Turing formula. As this formula cannot be applied directly in many circumstances, I gave a related method, generalized absolute discounting, that can be applied in similar situations as the Good/Turing formula, but which is supposed to give more robust estimates.

I showed how a morphological description given in form of a finite state transducer can be turned into a stochastic model for the generation of words, how this can be used for morphological disambiguation, and how such a model can be made more robust, so that it can produce

or accept arbitrary strings with low probability.

I sketched a method how the parameters of a morphological model can be trained from data that contains errors, and how models for word probabilities can be adapted to domains that are different from the source of the training data.

Some of the techniques given in this chapter are mainly intended as building blocks, that can be used as modules in a larger system or further generalized to the more interesting cases in which the estimates are influenced by contextual information.

Chapter 6

Bigram Probabilities

6.1 Motivation

As already explained in Chapter 4, the estimation of bigram probabilities can serve as a central building block for many different stochastic models for linguistic structures. As soon as we generalize the classical notion of bigrams, based on adjacent tokens in the input string, to pairs of lexical items that stand in certain linguistic relations, it becomes clear that these can be incorporated into many different approaches to grammar, and hence are useful to construct stochastic language models out of such grammatical theories.

The details of how such an enrichment can be performed still depend somewhat on the theory to be enriched, and cannot be discussed in full generality.

The basic idea is to exploit dependency relations between lexical items to assign a probability to any possible analysis of an utterance. We assume that an analysis provides a hierarchical organization of the concepts involved, and we try to model the generation of an utterance via a stochastic process (more precisely: a Markov branching process) that constructs the dependency structure of the utterance, step by step, where the probability of each individual step is conditioned on a small amount of contextual information, mainly consisting of the immediately dominating predicate. Models of this type have been heavily studied and described in recent literature

([Carroll and Charniak1992, Sleator and Temperley1991, Collins1996, Carroll and Rooth1998, Eisner1996, Chelba and Jelinek1998]), and it has been shown in [Johnson1996] that a similar model can be integrated into an LFG framework. Linguistically, these models are related to grammar formalisms that describe the dependency structure between words of an utterance. This idea (actually a commonplace) is the core of Dependency Grammar, but also shows up in Categorial Grammar, HPSG, and LFG. Examples for this will be shown in later chapters. This chapter will concentrate on the "pure" problem of estimating the probability of a composite linguistic event $\langle a, b \rangle$ from frequency data on such events.

As we have seen in the last chapter, even the estimation of word probabilities suffers from a considerable sparseness problem. If we now switch to composite events from a much larger domain, it is obvious that smoothing gets much more important.

For the sake of concreteness, we will use frequency data taken from real text. Since we aim eventually at models of linguistically meaningful relations, we want to base our experiments on such data, which makes the collection of the necessary data somewhat more difficult, since an exhaustive, high-quality classification of linguistic relations occuring in large amounts of naturally occuring text requires linguistic tools that do not yet exist. In the trade-off between quantity and quality of training data, we choose a compromise by concentrating on data from one relevant relation, that of co-occurring adjectives and nouns. Such pairs can be extracted with existing tools (taggers) efficiently and with relatively high accuracy; both nouns and adjective are very large and informative domains that are subject to severe sparseness problems; they actually show a lot of interesting interdependencies, and we hope to find a stochastic model for bigrams that has considerably better prediction than a model based on the independence assumption. Last not least, this combination is so frequent, that extracting such pairs uses a substantial fraction of all tokens in a large corpus.

One application in which an estimate of the probability of an adjective/noun pair would be very useful is the translation of noun phrases of the part-of-speech shape `ADJ NOUN and NOUN` from English or German into a language like French, where adjectives are typically postponed. The adjective could either refer to the first noun or to the conjunction

of both nouns, and the translation depends on this choice[1]. To find the right translation for *"religious belief and practice"*, it is important to know whether *"religious practice"* is a plausible noun phrase, i.e. to estimate its probability. As *"religious practice"* appears 41 times in the BNC, and as *"practice"* alone is much more general than *"religious belief"*, the grouping *"religious [belief and practice]"* appears to be more probable than *"[religious belief] and practice"*. Apart from MT, there are obvious decoding applications, where one of the words cannot be recognized with certainty by an OCR or speech recognition system. In such a situation, it might for instance be necessary to decide between *"abandoned farm"* and *"abandoned form"* based on probability estimates for adjective/noun pairs.

However, things are not always that simple. Collecting adjective/noun pairs from the BNC, we can easily see the extent of the sparseness problem. Using tools available at IMS, I extracted 5096635 pairs, which, after lemmatization, gave rise to 1563742 different types. 1086313 of them appear exactly once in the corpus. An application of the Good/Turing formula shows that the expected ratio of unseen pairs is more than 21%. In contrast to this, only 53262 noun lemmata and 67611 adjective lemmata appear exactly once in this data, giving rise to an estimated "failure rate" of 1% and 1.3%, respectively. Obviously, finding good probability estimates for unseen pairs will be crucial in order to achieve reasonable results in the general case.

6.2 Overwiew of Published Work

There is ample literature related to the subject, so that the following discussion must stay far from being exhaustive. The publications can be distinguished along various criteria:

- Goals
  - optimizing probability estimates

[1] One can avoid a decision by swapping the order of the nouns in the translation, but this does not always work, as the gender of the target adjective might depend on its scope, and there is also a certain risk to distort nonlocal references, e.g. if the next sentence goes on to compare *"the former"* vs. *"the latter"*.

  - finding meaningful clusters and similar words
  - reducing resource consumption of model

- Scope of problem and background
  - statistical work on estimation from large, sparse contingency tables
  - work on stoachstic language modeling based on adjacent n-grams
  - work on nonadjacent bigrams

- Approaches and methods
  - based on Good-Turing formula
  - mixtures of bigram and unigram estimates
  - based on clustering
    * hard clusters
    * soft clusters
  - based on similarity
  - dimensionality reduction

The goals are of course somewhat interrelated, but there is also a trade-off between them. For instance, finding and exploiting meaningful clusters may help to improve probability estimates and compactness of the representation to some extent, but since almost any assumption of independence can be refuted with samples of enough size, one cannot expect optimal prediction from a model that is heavily based on clusters and the underlying independence assumptions. Methods based on mixtures of fine-grained and coarse models usually outperform both of their ingredients, but this gain in performance has its price in some computational overhead, since both the fine-grained and the more general model have to be stored and used.

Some articles do not only describe specific methods, but actually compare the performance of various alternatives on the same datasets on one or several of the criteria given.

Given the difficulty to classify the papers, I will organize them in groups of similar papers, which I put approximately in temporal order.

The earliest treatment of the problem I have looked at is given in [Good1956].

Good describes a method that is applicable to tables where the assumption of independence has been rejected, and where there is no natural way to collapse (lump) rows and columns according to their similarity. In passing, Good defines a method for "weighted lumping", based on correlations between rows and columns, but admits that there are many related variants, and that more research would be needed to decide between them. The method described in the core of the paper is based on the assumption that the probability of a joint event $\langle s, t \rangle$ can be written as a product of the probability under the assumption of independence and an association factor $x_{r,s}$ that depends on the cell.

$$p_{r,s} = p_{r.} p_{.s} x_{r,s}$$

$x_{r,s}$ is assumed to behave according to some prior distribution (Good gives two possible priors and shows how to estimate their parameters from the data), and expectations for actual cell values are estimated in a Bayesian way by integration over all possible values for $x_{r,s}$.

The method is demonstrated with two examples. One consists of a table that associates occupations of fathers and sons (14 x 14 professions), the other is based on a larger table that associates occupations with causes of death, from which Good manually selected 18 representative occupations and 13 causes of death to do his calculations. Due to the concentration on frequent events, the data is much less sparse than typical language data, so it is questionable how well the method would work for our applications.

Good performs the calcualtions using two different prior distributions (one a log-normal, the other a Pearson distribution of the third kind), and stresses that the results are not very much sensitive to the choice of the priors.

The result of his computation can be seen as a "compromise" between the observed value and the expected value under assumption of independence, and depends only on these two values. For instance, in one table he estimates an expectation of 3.3 for an unseen event, for

which the independence assumption would lead to an expectation of 5.

Given new estimates for the cell probabilities, the parameters of the distribution of association factors can be reestimated, which however leads only to minimally different values in the examples he shows.

In [Good1964], ideas on weighted lumping are given in somewhat more detail. The general formula for expected cell frequencies is given as

$$\frac{\sum_{i',j'} \left(\frac{n_{i'j'}}{N} \frac{n_{i.} n_{.j}}{n_{i'.} n_{.j'}} f_1(\rho_1(i,i')) f_2(\rho_2(j,j')) \right)}{\sum_{i'} f_1(\rho_1(i,i')) \sum_{j'} f_2(\rho_2(j,j'))}$$

where $\rho_1(i,i')$ and $\rho_2(j,j')$ are sample product-moment correlation coefficients between pairs of rows i, i' and pairs of columns j, j', respectively, and where f_i and f_2 are increasing functions of their arguments. Good states that these functions should depend on the observed frequency $n_{i,j}$ in the cell to be estimated[2] But without concrete proposals for the choice of the functions f_x and without any empirical investigation, the discussion stays somewhat abstract and speculative.

Another interesting thought pursued in [Good1964] is the use of singular value decomposition of the contingency matrix, an idea that has recently attracted many reasearchers in information retrieval and NLP [Deerwester et al.1990, Schütze1993, Schütze1992]. Good gives many interesting mathematical insights, but does not address the problem that an approximation of a frequency matrix by a rank k matrix that minimizes the euclidean distance to the observed data will lead to negative cell estimates, which are not useful as estimates of frequencies.

A method for smoothing bigram probabilities that has been very popular in speech recognition applications is that of a linear mixture of various estimators. In speech recognition, the task of language modeling is usually formulated as that of estimating the *conditional* probability of some word w_i given some left context in a short window, which usually does not exceed two words, i.e. $P(w_i|w_{i-2}, w_{i-1})$. N-gram statistics derived from training data leads to three three natural estimators

[2]Being more liberal in the correction of small values than in large ones is intuitively appealing, but can lead to inconsistent estimates. Good does not discuss how to avoid this problem.

for this probability, based on relative frequencies. We may take the relative frequency of trigrams, given the bigram

$$f(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

or we might ignore certain parts of the available information on the context and use

$$f(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

or even

$$f(w_i) = \frac{C(w_i)}{N)}$$

where N is the number of words in the sample. These estimators have in a way complementary weaknesses. $f(w_i|w_{i-2}, w_{i-1})$ will in the limit give the most accurate estimates, since ti takes more context information into account, but it has so many parameters that there will rarely be enought training data for all of them. The other estimators are less informed and hence worse in the limit, but they do not suffer that much from the problem of sparse data, so that they may provide valuable distinctions among unseen trigrams. In a long series of papers [Jelinek1976, Jelinek and Mercer1980, Nadas1984, Nadas1985, Katz1987, Jelinek, Mercer, and Roukos1992], the IBM speech recognition group has produced many interesting ways to combine these estimators. A general form of this combination is given in [Brown et al.1992] as

$$\begin{aligned} P(w_i|w_{i-2}, w_{i-1}) &= \lambda_3(w_{i-2}, w_{i-1}) f_3(w_i|w_{i-2}, w_{i-1}) \\ &+ \lambda_2(w_{i-2}, w_{i-1}) f_2(w_i|w_{i-1}) \\ &+ \lambda_1(w_{i-2}, w_{i-1}) f_1(w_i) \\ &+ \lambda_0(w_{i-2}, w_{i-1}) f_0 \end{aligned}$$

where the f_i are relative frequencies based on i-grams and the λs are mixture factors (with $\sum_i \lambda_i = 1$) that are based on the frequencies $C(w_{i-2}w_{i-1})$.

The fact that the mixture coefficients depend on the frequency of the context allows quite some flexibility in applying this kind of mixture.

Intuitively, the mixture should give more weight to the trigram estimate in frequent contexts where this seems well estimated, but should fall back to bigram or even less informed estimates in less frequent contexts. Details of how the mixture can be optimized using a cross validation technique (they call it deleted estimation) are given in [Jelinek1998].

The variant of this flexible mixture described in [Katz1987] makes explicit use of the Good/Turing formula as described in [Good1956] to estimate the relative contribution of the various parts of the model. The approach has been widely used and is usually referred to as "back-off" model.

If we restrict attention to the case of bigram estimates, and apply it to estimate the joint probability of two events $P(A, B)$, this approach essentially boils down to a flexible mixture of a model derived from the joint distribution of both events $f(A, B)$ and the marginal distributions $f(A)$ and $f(B)$.

A more recent enhancement to these models is described in [Brown et al.1992]. Here, the intuition is that distributional similarities among words can be used to group similar words into classes and use this classification to derive estimates that suffer less from the problem of sparse data. The idea is to assign each word w to some class $C(w)$ and to assume that the probability $P(w_i|w_{i-1})$ can be written as

$$P(w_i|w_{i-1}) = P(w_i|C(w_i))P(C(w_i)|C(w_{i-1}))$$

which essentially contains an assumption of conditional independence between adjacent words, once their class is given. This assumption is clearly wrong in some cases, and any nontrivial classification will lead to fitting the training data somewhat less than a model based on words. However, the hope is that a classification that gives a relative good fit to the data will incorporate some useful generalizations, which might help to predict unseen events.

The authors concentrate mainly on the practical question of how to find a classification that maximizes the mutual information between both components. There is, however, no known algorithm that would solve this problem in polynomial time, so any implementation of such models must be based on some approximations. The authors use a greedy algorithm that successively merges those pairs of classes for

which such a "merger" results in a minimal loss of mutual information. Even this approximation would involve a computational complexity of $O(V^5)$ operations, where V is the size of the vocabulary. The authors have found a way to reduce this to $O(V^3)$ operations by clever reuse of intermediate results, and they apply it to very large vocabularies in a way where merging of clusters and incorporation of new words (in frequency order) are interleaved, to achieve a further reduction in computation overhead. They mention in passing that this agglomerative clustering yields more information by providing a classification tree and hence a kind of similarity measure over the entire vocabulary.

The model that results from this classification is bound to lose many idiosyncatic distinctions involving specific words, and even if one can get a somewhat better generalization power in return, it is not to be expected that a model based on a relatively strong independence assumption can fit the data as well as a model based on words. However, it turns out that a mixture of a class-based model with a word-based one can be better than both of them alone, and the authors were able to achieve a 3.3% reduction in perplexity, compared to the model without classes.

A systematic and very detailed evaluation of various estimators of bigram probabilities is given in [Church and Gale1991]. It is based on a corpus of about 44 million words taken from the 1988 AP news wire, which the authors randomly split into two halfs for training and testing purpose. All the methods they compare try to estimate an expected cell frequency based on the observed frequency $r = f_{ij}$ and on the expected frequency under assumption of independence $jii = f_{i.}f_{.j}/f_{..}$. They compare the following estimators: STD: the "gold standard", trained on all available data (including test data), MLE: the maximum likelihood estimator based on f_{ij}, without smoothing, BGT: basic Good/Turing smoothing applied to f_{ij}, ignoring the marginals, UE: unigram estimator based on the marginal alone, EDE: enhanced estimator using deleted estimation and EGT: enhanced estimator using the Good/Turing formula. Here, the enhanced methods are based on a grouping of the data into bins according to the second predictor, and using deleted estimation or the Good/Turing formula to find optimal prediction for each group.

They do not only evaluate some overall figure of merit such as the perplexity, but instead show plots for various values of f_{ij}, where the estimates of the methods are plotted against jii. Since these plots also show the values of STD, which would be optimal under the given circumstances, it is relatively easy to read systematic problems of the estimators from the plots, and also get an impression of some arbitrariness visible in random fluctuations of the STD estimator.

The result of their study is essentially that deleted estimation and Good/Turing estimators show comparable performance, with a slight advantage for the Good/Turing approach. Not surprisingly, the enhanced methods perform better than their basic counterparts.

Although this study provides a lot of very valuable insights, there are some aspects to their method that make it look less fortunate as a basis of a practical system. Their application of the Good/Turing formula assumes that the frequency data is itself smoothed, before being entered into the formula. They use a smoother by Shirey and Hastie (1988), but they do not give a reference, and I was not able to locate the details. Also, grouping of the cells according to the expectation under assumption of independence looks somewhat tedious and does not automatically incorporate certain plausible constraints on the monotonicity of the estimators (e.g. in their approach, for a given observed frequency, a higher value of jii might lead to a smaller estimate). Last not least, it is questionable whether the product of the marginal distributions is the best additional evidence we could use for refining the basic methods. From the bigram data we can easily extract information about the spread of the distribution per row and per column, and it looks unwise not to use such information. Also, the autors do not consider the use of corpus-derived similarity information to get more reliable estimates of the frequencies.

Another group of papers on the subject has been published by the group of Hermann Ney [Ney and Essen1993, Ney, Essen, and Kneser1994].

In [Essen and Steinbiss1992], the authors investigate the idea of smoothing bigram probabilities using information on co-occurrence. From bigram frequencies, they derive confusion probabilities, which they use as a similarity measure they use for smoothing. Apparently, the results were not as promising compared to to the implementa-

tional difficulties, so that this group did not continue the work on co-occurrence based smoothing.

Pereira has co-authored a set of papers that explore various approaches to soft clustering and similarity-based smoothing. In [Pereira, Tishby, and Lee1993], the authors set up a soft classification scheme in which pair of events is assumed to be generated by a hidden class. The classification is probabilistic, i.e. each hidden class has probability distributions over the two components of the pair, which are assumed to be independent, given the class. We have

$$p(a,b) = \sum_c p(c)p(a|c)p(b|c)$$

Since the training data does not specify the classes that may have generated it, they employ a divisive clustering procedure, in which, starting from a unique cluster, random splits are investigated and the split that achieve the best reduction in training set entropy is chosen, until a specified number of clusters is reached. As expected, they observe a tradeoff between generalization cabability (which is optimal for a small number of clusters) and the flexibility to fit the data, which increases with the number of clusters, at the price of overfitting the data.

[Rooth1995] uses the same underlying idea of probabilistic classification, but employs the EM-algorithm in a more or less straightforward way to achieve the classification, starting from randomly initialized distributions. My own experimentations with this setup lead me to the conclusion that the variation speed of the parameters is very heterogeneous, in the sense that a large number of iteration is needed until the high frequency events are sorted into reasonable classes, whereas the low frequency events show a drastic overlearning effect even after few iterations and even for small number of clusters.

In [Dagan, Pereira, and Lee1994], the authors base the estimation of bigram probabilities on a dynamic combination of statistics from similar words, where the similarity is defined using the Kullback-Leibler distance, an information-theoretic distance measure on the empirical distributions. [Lee1997] is a somewhat more systematic exploration of this idea, comparing a set of distance measures (including the confusion probabilities of [Essen and Steinbiss1992]).

Finally, [Hofmann and Puzicha1998] gives a rather extensive overview of methods based on soft clustering, which are also put in a somewhat broader context.

I also want to mention contributions by [Grefenstette1994] and [Schütze1992, Schütze1993], which, although not directly aimed at the estimation of co-occurrence probabilities, show that a systematic exploration of co-occurrence patterns in textual corpora can reveal very interesting semantic similarities between words. Grefenstette shows how such similarities can be used to speed up the construction of domain-dependent semantic resources, whereas Schütze uses SVD as a technique for dimensionality reduction of the co-occurrence relations for a broad variety of applications, including information retrieval.

It is interesting to observe that formally, the matrix decomposition that is done in SVD is closely related to the probabilistic decomposition based on hidden classes used by [Pereira, Tishby, and Lee1993] and [Rooth1995].

6.3 Generalizing Absolute Discounting to the Two-Dimensional Case

As we have seen in the last chapter, absolute discounting, i.e. subtracting a small constant from all observed counts and redistributing the saved probability mass in some proper way can lead to quite reasonable estimates. In this section, we want to explore to what extent this method can be applied to two-dimensional contingency tables. We want to use some form of absolute discounting to gain some probability mass which we then redistribute in some plausible way on the cells. To give some idea of what will follow, we might first make the simplifying assumption that we do not want to modify the marginal distributions and assume somewhat implausibly that the amount of probability redistributed to a cell is proportional to the product of the probabilities collected form the cell's row and column. This leads to the following formula[3]:

[3]In the sequel, we will use the following notation. We assume that cell frequencies are given as $f(a, b)$, where a and b are linguistic events. For any $i \in N_0$, we define

$x(a,b) = f(a,b) + d * (t(a,\_) * t(\_,b)/t - t(a,b))$

A nice property of this formula is that there is only one parameter d to estimate. We can employ the Good/Turing formula to find out how much probability mass should be redistributed on empty cells and use this to set an appropriate value for d. However, we need to compensate for probability that flows back to nonempty cells. Before we go into details of how to estimate the parameters, we should try to find a more general formula that addresses the problems that have been mentioned above. Redistribution proportional to the product $t(a,\_) * t(\_,b)$ may not give the right balance between frequent and infrequent parts of the table, so we would like to have a component that depends only on some row statistics or on some column statistics, but not on the product of both. We also might want to have some contribution spread uniformly over the whole matrix.

Whereas absolute discounting seems to be a good heuristic for a rough approximation, in many linguistic datasets we can observe that the application of the Good/Turing formula results in a different discounting constant for event types seen once than for those seen more than once. We can make the mixture flexible enough to accomodate for this effect by treating events seen once differently from those seen more than once.

An estimator that is general enough to cover all of these phenomena can be defined as follows.

$$
\begin{array}{rrrrr}
x(a,b) = & \kappa_1 n_1(a,b) & +\kappa_2 m(a,b) & +\kappa_3 r(a,b) & \\
& +\mu_{00}1 & +\mu_{01} n_1(\_,b) & +\mu_{02} m(\_,b) & +\mu_{03} r(\_,b) \\
& +\mu_{10} n_1(a,\_) & +\mu_{11} n_1(a,\_) n_1(\_,b) & +\mu_{12} n_1(a,\_) m(\_,b) & +\mu_{13} n_1(a,\_) r(\_,b) \\
& +\mu_{20} m(a,\_) & +\mu_{21} m(a,\_) n_1(\_,b) & +\mu_{22} m(a,\_) m(\_,b) & +\mu_{23} m(a,\_) r(\_,b) \\
& +\mu_{30} r(a,\_) & +\mu_{31} r(a,\_) n_1(\_,b) & +\mu_{32} r(a,\_) m(\_,b) & +\mu_{33} r(a,\_) r(\_,b)
\end{array}
$$

In this formula, $m(a,b)$ stands for $t(a,b) - n_1(a,b)$, which is 1 for cells with frequencies larger than one, and $r(a,b)$ is defined as $f(a,b) - t(a,b)$,

$n_i(a,b)$ as 1 if $f(a,b) = i$, and 0 otherwise. We define $t(a,b) = \sum_{i=1}^{\infty} n_i(a,b)$, i.e. $t(a,b) = 1$ iff $f(a,b) > 0$. We write $f(a,\_)$ as shortcut for $\sum_b f(a,b)$, $f(\_,b)$ for $\sum_a f(a,b)$ and f for $\sum_{a,b} f(a,b)$. $n_i(a,\_), n_i(\_,b), n_i$ and $t(a,\_), t(\_,b), t$ are defined analogously.

i.e. it is the frequency reduced by one if it was positive. The κs and μs are supposed to contain the normalization constants, i.e. all coefficients are non-negative and set up in such a way that $\sum_{a,b} x(a,b) = 1$.

This estimator is a mixture of 17 components, i.e. we have an optimization problem with 16 free parameters. A natural criterion to maximize is cross entropy on unseen data. In order to achieve maximal exploitation of the available data, we can train the mixture of the components using a combination of the EM-algorithm with cross-validation or leaving-one-out training.

6.4 Exploiting Similarities

In order to make best use of similarities between rows and/or columns to get more reliable probability estimates, we need to address a couple of questions.

- What similarity measure should we employ?
- How to find the best amount of smoothing from a given similarity value?
- How to control the computational implications of such an approach?

Many similarity measures have already been investigated for the task of smoothing. Prominent examples are correlation coefficients[Good1964], confusion probability[Essen and Steinbiss1992], cosine and euclidean distance[Schütze1992], and various distance measures inspired by information theory[Lee1997].

Usually, some high enough similarity between certain rows or columns is taken as a justification to mix the estimates from these rows or columns to some degree that depends on the similarity. If two rows are to be smoothed against each other, usually some fraction of the frequencies of these rows are pooled and the estimates from this pool are then again redistributed. This technique is in a way analogous to linear discounting, since high counts are subject to more modification than low counts. This is problematic when applied to natural language

frequency data, in which collocations and idiomatic expressions play an important role. Such combinations usually do not generalize to similar words, even if many other partners can be interchanged freely. If the goal is to achieve high accuracy in the probability estimates, we should not use high counts for smoothing at all. In the sequel, I want to pursue an alternative that does not have this problem. The main idea is to apply smoothing only to a nonlinear fraction of the observed frequencies, for instance by taking only that part of a count that has been discounted.

A related argument applies to the choice among similarity measures. Any such measure that concentrates on high frequencies might miss important similarity structures which are hidden behind very frequent idiosyncratic combinations. It seems therefore more more plausible to use similarity measures that concentrate on those parts of the distributions that are most relevant for smoothing, i.e. their tails. This argument rules out most of the distance or similarity measures mentioned above, since they are all dominated by high frequencies.

Of course, such an intuitive argumentation needs to be validated empirically. One approach for the identification of the best similarity measure would be to implement some of them and to determine empirically which one gives the best results in the prediction task. This is essentially the idea behind the work described in [Lee1997]. Apart from being somewhat tedious, this approach has the disadantage that the optimal similarity measure might not be in the set that has been implemented. Therefore, I want to explore a somewhat different strategy, which takes the motto "let the data decide" to an extreme. We can set up some space of smoothing schemes in parametric form and use an unsupervised training procedure to find a set of parameter values that is optimal for the given task and data. However, such an approach requires great care to avoid problems with overtraining. It is therefore not very wise to set up a model that allows free exchange between arbitrary pairs of rows and/or pairs of columns, and hope that the training procedure might find the pairs for which smoothing is most important. Such a model has far too many parameters to be trained with any amount of available training data. Instead, we can try to identify some simple statistical indicators of similarity and define the smoothing scheme in such a way that more similarity will result in more smoothing. If there

is a choice between the indicators, or if it is unclear how much smoothing should be applied for a given similarity level, we can try automatic training procedures to optimize these details empirically.

For any given pair of rows or colums, we can collect statistics on co-occuring events and use this to find some indication on the amount of smoothing that would be appropriate among this pair. One possible measure is the Jaccard coefficient[Romesburg1990], that relates the intersection of the sets of possible partners with the union. This ignores all frequency information (apart from the distiction 0 vs. positive), and is therefore a good candidate for a measure that does not get confused by high frequencies. However, we might get a somewhat finer distinction by using frequency information in the following way.

If we concentrate on the situation in which smoothing is most important, namely the prediction of unseen events, we can simulate the generation of such events in a leaving-one-out setting as predicting events seen exactly once from data where just this observation has been removed. To generate a singleton event $\langle a, b\rangle$, we can assume that it has been generalized from seen events in the same row or column, i.e. events of the form $\langle a', b\rangle$ or $\langle a, b'\rangle$ with $a' \neq a$, $b' \neq b$ and positive f. In case where multiple possibilities exist we can divide the count equally among them. We can keep track of such generalization steps by accumulating fractional counts for pairs $\langle a', a\rangle$ and $\langle b', b\rangle$.

The "optimal flow" from row a' to row a is then given as[4].

$$flow(a' \rightarrow a) = \sum_{f(a,b)=1} \frac{t(a', b)}{t(a, \_) + t(\_, b) - 1}$$

Using this technique, we get quite detailed estimates for the optimal mutual flow of counts between rows and between columns, which we can then use to obtain much finer distinctions. Of course, finer distinctions are always subject to overtraining, and hence this method might sometimes fail to smooth out random sampling effects as well as a less

[4]This is only one of several possible variants. We might as well do the similarity calculation for rows and columns independently, we might also weaken the contribution of very rare events by adding a constant to the denominator or using some completely different weighting scheme like the logarithmic weights used in information retrieval [Grefenstette1994, Deerwester et al.1990]

finegrained method. But this situation is quite familiar. We often have to find compromises between finegrained but overtrained models and better estimated, but coarser models. We can assume that we can find a good combination using a linear mixture of these estimators, and as usual, we can apply the EM-algorithm to optimize this mixture.

A more severe problem is however the question of computational complexity and practical feasibility. If we perform the computation sketched above offline, we will have to store a confusion matrix for one or even two dimensions, which is much less sparse than the original data matrix. We can try to avoid this by computing and using the smoothing parameters on the fly for those pairs that show up in an application, but this implies summations that may not be feasible at runtime. A third possibility would be to compute the confusion matrix off-line, but prune it in such a way that only those pairs have to be represented for which the similarity is significantly above some general level.

6.5 Summary

In this chapter, I have given an overview of techniques that allow to estimate probabilities of word pairs. The intended application of these techniques is not primarily for bigrams of adjacent words, but for word pairs that stand in a certain linguistic relation, such as a dependency relation. I first give an overview of the most important published approaches to the problem. I then sketch how the method for estimating word probabilities from Section 5.4 can be generalized to the two-dimensional case, resulting in an estimation method that does not only use marginal frequencies, but also somewhat more information on the scattering in the rows and columns, which can be derived from marginal statistics on types and singletons. Finally, I shortly discuss approaches that determine and use similarity measures between rows and between columns. I try to motivate why the measures of similarity that are typically used in such approaches are not optimal for this task and sketch possible alternatives.

Chapter 7

Using More Context

7.1 Introduction

Stochastic language models based on word n-grams make rather strong independence assumptions, and hence are very selective in the use of context for their predictions. This has been the target of heavy criticism. Grammar-based language models are based on different independence assumptions, which still seem to exclude many important sources of evidence. In this chapter, I want to discuss approaches to overcome this limitation.

Context ignorance in simple generative models

So far, the stochastic models typically used in speech recognition are based on trigrams of adjacent words. Superficially, these models use more context information than a grammar-based bigram model, but this can only partially compensate the problem that the type of context they use is often not a all informative. For instance, a trigram model that is to predict the noun *hole* in the sequence *"drill a small hole"* does not know about the verb *drill*, because it is already too far to the left. The model has to rely on the context *a small*, which is so general that the probability of *hole* in this context is not particularly high.

The obvious presence of dependencies over longer distances has been the standard argument against these models made by many linguistic researchers, in the tradition of [Chomsky1957]. In

[Chomsky and Miller1963], the authors give as an example the sentence *"The people who called and wanted to rent your house when you go away next year are from California"*, where the plural subject *people* and the plural verb *are* stand in an agreement relation. They then go on to explain that a stochastic n-gram model that was able to capture this kind of dependency would lead to an astronomic number of parameters, which would have to be learned from data, a model which is ridiculously unrealistic.

Despite these principled problems, some research in language modeling for speech recognition went into n-gram models with $n > 3$ [O'Boyle, Owens, and Smith1994]. This had some limited success in restricted domains, but in the general case, the number of possible n-grams gets so huge that the chance of reusing a larger context *as is* is practically zero, and the extension of the context in this crude way does not improve the quality of the model. A different approach that has been pursued tries to exploit the fact that people tend to reuse recent words in their utterances. [Kuhn and De Mori1990] describes a cache-based language model in which the probability of words that have been uttered previously is increased. However, in practice, this effect is not limited to repetition of identical words, but affects also words that stand in a semantical relationship. Such effects cannot be easily accounted for in a cache-based model.

Reasons like this have motivated work on the incorporation of stochastic models into structurally richer grammatical formalisms such as PCFGs. Grammar-based bigram models are able to exploit context up to an unlimited distance, and they are able to combine evidence found in this context in an interesting way. In the example given above, we can assume that both the presence of *drill* and of *small* in the context increase the probability of *hole*, i.e. $p(\text{hole}|\text{drill a small}) > p(\text{hole}|\text{ a small})$ and $p(\text{hole}|\text{drill a small}) > p(\text{hole}|\text{ a } \langle\text{ADJ}\rangle)$, and a grammar-based language model is able to exploit such cumulation of evidence, even in cases where additional material (e.g. adjuncts) of arbitrary length comes between the verb and the object noun.

But also context free models make very strong independence assumptions, and even if they are lexicalized as in the case of the bilexical grammars [Carroll and Charniak1992, Sleator and Temperley1991, Collins1996, Carroll and Rooth1998,

Eisner1996, Chelba and Jelinek1998], the amount of context information they can support is still rather limited. It is the characteristic property of context-free models that the subtree below a node with a given category depends only on that category, and nothing else. In the lexicalized models, the category is enriched with the head lemma of the constituent, which can be taken as additional information on which to conditionalize the generation of modifiers. But we still have the situation that the annotation of this node is a kind of "bottleneck" through which all dependencies between the local subtree and the context of it have to be mediated.

One way to phrase the problem is that the amount of context information that can maximally be used by a model based on a context-free structure is limited by the information that can be encoded in one category symbol.

There are important kinds of dependencies that cannot easily be represented in such a context-free manner. The idea that subject and object of a verb are generated independently after the generation of the verb, and that all interdependencies between them have to be communicated through the verb is a strong simplification of the facts. This is particularly striking in cases where the verb expresses a very general semantic relation, such as "contain", where the assumption of independence between subject and object would lead to a large probability for implausible combinations. We could of course try to find a special treatment of such cases that propagates more information through the structure and assume that the generation of complements is not only conditioned on the head, but also on the complements that have already been generated. But even if we take a much richer notion of local context into account, there are long-term effects, such as theme, discourse structure, genre and individual style which all influence the lexical choice. I think it will be very difficult to find any pair of properties $\langle A, B\rangle$ of natural language utterances that can be shown to be stochastically independent, or any triple $\langle A, B, C\rangle$ of such properties, which form a Markov chain, i.e. for which conditional independence of A and C, given B can be shown.

In the light of this conjecture, the goal of simple stochastic language models based on bigrams can not be to build a realistic model of language performance. It should be rather seen as providing the closest

approximation that is possible under a set of constraints that guarantee a certain regularity and simplicity.

There are some arguments that can be made in favor of such impoverished models. We might for instance argue that a conventional (i.e. non-lexicalized) PCFG is a suitable way to express *syntactic* preferences, with other words, we could take the expressive power of a certain formalism as a more or less natural way to separate different descriptive levels. But I do not think that this argument is valid, because even syntax is usually and most naturally described in more expressive formalisms, like constraint-based grammars. Furthermore, simple syntactic preferences, like the dispreference of center-embedding or certain attachment preferences cannot easily be incorporated into a context-free grammar. Restrictions of the formal power of a grammar formalism can be circumvented by increasing the number of the symbols that are allowed, so they do not seem to be the right way to separate the linguistic levels.

Another, more practical argument in favor of stochastic models based on random processes is that the optimal setting of the parameters can be determined based on relative frequencies, if we ignore the sparseness problem and assume access to an unlimited amount of training data. As we will see below, this property does not hold for more general ways of combining contextual evidence.

7.2 Stochastic Models of Richer Interactions

If our goal is to optimize the predictive power of a theory, given limited resources, we should also look at ways to represent the relevant information more compactly. Whereas CFGs are based on an unstructured vocabulary of atomic categories, constraint-based grammars use feature vectors as their representation, which do not only allow for a more compact expression of generalizations, but which are also able to store more information in the categories. It should be possible to use this expressive power to build language models that give better results with a description of fixed size.

In the literature there are several approaches that try to use stochastic knowledge from multiple descriptive levels and combine it into one predictive model. One of the earliest of them is [Mark et al.1992], where the authors multiply the probability of a given string under a SCFG model with a score that depends on the adjacent word bigrams that appear in the string. The problem with such a combination is that it is not obvious how to set it up to give a model that is overall consistent, i.e. generates a string with probability 1. To determine the parameters for this model, the authors apply a sampling procedure based on the Metropolis algorithm that allows to generate a random sample according to a given distribution of this form, which is then used to iteratively adjust the bigram scores to improve the model.

This procedure is an application of a more general method to train parameters for a Markov Random Field, which is a very popular approach to model prior distributions on bitmaps in image processing ([Geman and Geman1985] see also [Winkler1995] for an overview). The general idea behind these models is to define a probability distribution over possible configurations in terms of certain properties of such configurations, where the presence of such properties make a configuration more or less likely.

If we look at the problem in a more general way[1], we can see it as that of estimating the cell probability in a high-dimensional contingency table, which is studied in multivariate categorical data analysis [Agresti1990]. A standard approach is then to use log-linear models, in which the probability $P(\langle e_1, \ldots, e_n \rangle)$ of some vector of events is modeled as the product of several terms, each of which depends only on some proper subset of the vector. [Franz1997] has used log-linear models for the resolution of attachment ambiguity with considerable success.

[1] And ignore for a moment the structural information that is conveyed in the context

7.3 Augmenting a Constraint-Based Grammar with Probabilities

A proposal that is closer to the idea of constraint based grammars was given in [Eisele1994]. The original idea was to attach probabilities to the context-free skeleton of a constraint-based grammar specified in the CUF formalism [Dörre and Dorna1993]. But as nothing in that paper was specifically related to CUF, we could also assume a definite clause grammar or any other formalism that allows to enrich a context-free description with features. The idea behind the proposal was that a logical program with probabilistically weighted clauses defines, analogously to a PCFG, a random branching process for the generation of proof trees. Different from the context-free case, the nodes in the proof tree are annotated with first-order terms in the DCG case or with constraints on feature values in the CUF case. These annotations can make proof trees inconsistent, so that the model does not generate a consistent tree with probability 1[2].

In order to obtain a probabilistic model over all proof trees that are consistent with a given goal to prove, I proposed in [Eisele1994] to renormalize all the resulting probabilities by their sum. One might wonder why this technical trick should be useful to introduce context-dependencies that cannot be expressed in the original branching process, but actually, it allows for the expression of arbitrary log-linear models, which form a very general class of statistical models.

Assume that `p(A,B,C)` is a ternary relation over atomic values that is to be modeled in a logic program with weighted clauses[3]. According

[2]PCFGs do not necessarily define probability distributions over the set of all the finite trees (and hence strings) they generate. But in the context free case, the possible loss of probability mass is a somewhat exotic marginal case, that can be easily detected [Booth and Thompson1973] and which does not appear if the parameters are trained from data [Sánchez and Benedi1997]. In the probabilistic extension to CUF, it is a central property that cannot easily be avoided.

[3]Here and in the sequel, we will drop the requirement that the probabilities for all clauses of a predicate sum up to 1. We will, however, assume that for all goals, the sum of the weights of all possible solutions converges. This can be guaranteed by requiring that no goal can have infinitely ambiguous solutions or that all clauses involved in such infinite enumerations have weight < 1.

to the kind and strengths of the independence assumptions we want to make, we can define it in one of the following ways, where the predicates on the right-hand sides are assumed to be defined in an enumeration of weighted facts:

- independence
  `p(A,B,C) :- q(A), r(B), s(C).`
- partial independence
  `p(A,B,C) :- q(A), r(B,C).`
- conditional independence
  `p(A,B,C) :- q(A,B), r(A,C).`
- no-three-way interaction
  `p(A,B,C) :- q(A,B), r(A,C), s(B,C).`
- saturated
  $\texttt{p(a}_1\texttt{,b}_1\texttt{,c}_1\texttt{).}$
  `...`

The augmentation of a DCG with probabilities, or vice versa the augmentation of a SCFG with features, looks like a simple and straightforward synthesis of these concepts. However, it has a rather dramatic effect on the way the parameters for such a model have to be determined. In [Eisele1994], I gave the following simple example to highlight the problem.

$\texttt{s(X)} \leftarrow_1 \texttt{p(X), q(X).}$

$\texttt{p(a)} \leftarrow_{p_a} .$
$\texttt{p(b)} \leftarrow_{p_b} .$
$\texttt{p(c)} \leftarrow_{p_c} .$

$\texttt{q(a)} \leftarrow_{q_a} .$
$\texttt{q(b)} \leftarrow_{q_b} .$
$\texttt{q(c)} \leftarrow_{q_c} .$

and the training data
`?- s(a).` (9 times)
`?- s(b).` (once)

If the standard method to find a maximum-likelihood model is applied to the unnormalized probabilities, the training algorithm will try to maximize $(p_a * q_a)^9 * (p_b * q_b)$ under the constraint that the p_i and q_i sum to 1 respectively. The optimal solution (without smoothing) to this will be $p_a = q_a = 0.9, p_b = q_b = 0.1, p_c = q_c = 0$, and that is what the training procedure sketched above will converge to. But this leads to a probability ratio between s(a) and s(b) of 81:1, instead of 9:1, as might perhaps be expected.

If one takes into account that the scores computed by the model are subject to normalization, the right value to maximize is $(p_a * q_a/(p_a * q_a + p_b * q_b + p_c * q_c))^9 * (p_b * q_b/(p_a * q_a + p_b * q_b + p_c * q_c))$, which means to maximize the probability of the training data relative to the cases where a consistent proof tree is built. This leads to different optimal values, e.g. $p_a = q_a = 0.75, p_b = q_b = 0.25, p_c = q_c = 0$, which reproduce the right probability ratio for s.

One can look at this problem as follows: Our model allows both predicates `p` and `q` to influence the probability distribution of the common variable `X`. Hence multiplying the probabilities leads to a model which, informally speaking, counts the evidence given in the training data twice. In [Eisele1994], I had to admit that the standard procedure for maximizing the unnormalized probability such a model assigns to the training data is not applicable, and I observed that the correct criterion to maximize is the normalized probability, which is more difficult because a modification of one of the parameters affects also the normalization constant, i.e. both the numerator and the denominator of the fraction to be maximized. The most severe obstacle in this situation is the fact that the denominator contains a sum over all consistent proof trees for the goals in the training data, which are typically not expressible in a closed form.

In [Abney1997], Steven Abney defines *stochastic attribute-value grammars* and shows that they can be seen as variants of a Markov Random Field. His way of attaching weights to the structures generated by the grammar is different from that of [Eisele1994], as he attaches the

weights not to program clauses that are used, but to nodes of the structures according to certain properties, which are given in form of a set of local subtrees. Actually, he assumes that the set of properties that can influence the probabilities of the configurations is not pre-specified by the grammar writer, but that it is determined dynamically, during parameter estimation, in a procedure that adds from a set of candidate properties one that gives a best improvement in the Kullback-Leibler distance between the model and the training data.

The question whether the properties to which the probabilistic weights attach should be selected by the grammar writer or by the training algorithm can not be decided theoretically as its answer might depend on the circumstances. Of course, automatic property selection is preferable, provided that it is actually able to identify the set of relevant properties. However, it could very well be that the grammar writer has an intuition about important dependencies in the data, which might be based on a more general view of the linguistic reality than anything an automated procedure might infer from a (possibly small) set of training data. In this sense, it is a more conservative approach to separate property selection from the training of the optimal weights of the properties, and assume that the former might be done manually, based on linguistic intuition.

No matter how this question is decided, the general term for the probability of a certain configuration x is given by

$$p(x) = \frac{\prod_i \beta_i^{f_i(x)}}{\sum_{y \in \Omega} \prod_i \beta_i^{f_i(y)}}$$

where β_i is the weight for the property i and $f_i(x)$ is the frequency of occurrence of property i in the configuration x. This is the form of a Gibbs distribution, which has been used in statistical physics and later as a solution to the problem of finding a maximum entropy distribution that follows a given set of constraints[Jaynes1995]. Models based on Gibbs distributions have been used for reasoning under uncertainty [Pearl1988] and in certain types of artificial neural networks called Boltzmann machines [Ackley and Hinton1985]. In computational linguistics, they have been popularized mainly by the group around the Della Pietras and used for

a variety of applications [Della Pietra, Della Pietra, and Lafferty1995, Berger, Della Pietra, and Della Pietra1996].

In all these models, the optimal values for the weights cannot be directly read off a set of annotated training instances, but have to be determined using random sampling. The purpose of this is to guarantee that the expected frequency of a property in a sample generated from the model is identical to the relative frequency of the property in the training data. If a random sample taken from a preliminary model reveals that this does not hold for a certain property, the weight of that property can be adjusted incrementally. [Riezler1998] gives a very nice overview on a probabilistic version of constraint logic programming that is based on Gibbs distributions. He gives a method for parameter estimation that combines properties of the EM algorithm with random sampling and is hence suitable for unsupervized learning, i.e. acquisition of the parameters from ambiguous training data.

The difficulty in parameter estimation may be one of the major reasons why the simpler, but contextually impoverished models based on generative stochastic processes are still very popular.

7.4 Summary

In this Chapter, I have discussed some properties of stochastic models based on generative processes, such as Markov models and stochastic context free grammars. I have given examples that show that these models cannot express stochastic dependencies that obviously exist in natural language. As an alternative, I have sketched a probabilistic variant of a constraint logic programming language that was first described in [Eisele1994]. This formalism has strong similarities with Gibbs distributions, maximum entropy models, and Markov random fields that have since been propagated in the literature. However, it also shares the problem that parameter estimation relies on random sampling, which makes the model somewhat difficult to use for expressing co-occurrence statistics on large lexical domains.

Chapter 8

Stochastic Ranking of LFG Analyses

8.1 Introduction

In this chapter, I want to present an experimental evaluation of a disambiguation system that is based on some of the concepts and techniques introduced in the preceeding chapters.

This evaluation is intended as a *feasibility study*, i.e. the goal is to demonstrate that these techniques can actually be helpful contributions to solve the disambiguation task. It does not try to achieve a fine-grained integration and optimization of these techniques, nor can the results obtained in this experiment directly compared to results in similar experiments that aim at the evaluation of the performance of general purpose parsing systems [Black and othersl1991]. I am convinced that the techniques presented here will eventually contribute to the construction of parsing systems that are, in some measurable sense, better than system that already exists. But the prove that this is in fact the case will require more work that is outside the scope of this thesis.

The experiment described in this chapter is based on the idea that the notion of stochastic language models can be applied in a framework based on Lexical Functional Grammar. [Johnson1996] describes a way to apply a generative stochastic model to the functional struc-

tures that play a prominent role in LFG. He sees the generation of natural language as a two step process, in which first a more abstract representation, the f-structure, is generated, which is then in a second step "translated" into a linear string.

This idea is appealing, because functional structures look like an appropriate level to express many of the weak regularities that can be observed in actual use of natural language. Hence a stochastic model that operates on this level should be able to exploit these patterns in a better way than models that operate directly on the level of strings. In a certain way, an LFG-based stochastic model tries to capture lexical co-occurrence patterns on a similar level as bilexical models like those in [Carroll and Charniak1992, Collins1996, Carroll and Rooth1998]. But the fact that the representations on which a model operates are based on an independently existing linguistic theory like LFG can be seen as a strong advantage, because this allows us to re-use the theoretical and implementational facilities that already exist. There are, however some drawbacks that are caused by the fact that f-structures and the corrsponding strings do not stand in a one-to-one correspondence or that are consequences of other problems of the current state of the art of LFG-based parsing.

- If one f-structure could be realized by several different strings, a stochastic model of f-structures does not lead to stochastic language model in the usual sense, which would be a model of the corresponding strings. This would make it difficult to compare the performance of different lanuage models in a straightforward waym e.g. by using cross-entropy. [Johnson1996] circumvents this problem by postulating that the f-structures have to encode enough information to make the generation of a string from a given structure deterministic.

- f-structures in general have the shape of a graph, which can contain re-entrancies and cycles. It is not obvious how to set up a stochastic model for the generation of such structures. The model of branching processes that underlies the usual PCFGs, even the lexicalized variants, does not apply.

- Since we cannot directly observe naturally occurring f-structures,

and the utterances that appear in text corpora are very often highly ambiguous, it is not obvious how we can collect the data to train the paramters of a stochastic model. But this problem is not specific to an LFG-based approach, as any stochastic model that assigns non-trivial structures to utterances will have to find a way to train the parameters from ambiguous training examples. A usual and staightforward way of learning from incomplete data this is the application of the EM algorithm, in which a preliminary model is used to obtain estimated frequencies of completed observations, which are then used to obtain improved estimates.

- Analyzing free text with a LFG requires a certain level of robustness both of the linguistic resources and the implementation of the parser, which can currently not yet been taken for granted. Also the need to analyze large text corpora to get sufficiently high counts imposes certain constraints on speed and robustness, that are currently somewhat hard to satisfy.

To cope with these problems, I have made a couple of simplifying assumptions that are theoretically not well motivated, but seem to approximate the real facts good enough to achieve useful results.

One of these compromises is the assumption that is is currently too early to use an LFG-based analysis for a general-purpose stochastic language model in the style of the models of the IBM speech recognition group, which have been trained on very large corpora[1]

Instead, we will test the disambiguation performance on relatively small set of manually disambiguated utterances, taken from a technical domain. Details of the evaluation method are described in the next section.

Another compromise is that I will not take some consequences of the graph-structured organization of f-structures into account, but will use a stochastic model of the creation of f-structures that is essentially a branching process in which the generation of a feature value is conditioned on the PRED-feature of one of the surrounding f-structures. I

[1] [Brown et al.1992] describes a model that has been trained on about 583 Million words of English text and then tested on the Brown corpus (about 1 Million words) for performance evaluation.

will even go further and ignore all syntactic features except the PRED and the syntactic relations that connect different levels of f-structures.

The reason for these inaccuracies is the difficulty to obtain sufficient amounts of training data that has been analyzed with an LFG. Instead, I will use more superficial ways of analysing free text, based on a shallow parser [Grefenstette1994, Ait-Mokhtar and Chanod1997], to obtain significant amounts of lexical training data. The hope is that such tools for shallow analysis are good enough to obtain statistical data that reflects the basic facts of language use and word co-occurrences, and that this information, although it has a certain error rate and does not contain all the fine details that would be relevant, are still useful for some baseline functionality. One can see the pred-pred relations as a least common denominator between an LFG-based and a shallow parser analysis, hence statistical information on such relations seems to be a suitable interface between the two worlds.

8.2 Evaluation Criteria

There are several ways to measure the performance of a stochastic disambiguator, and I will discuss some of the possibilities and motivate my choice.

If ambiguous analyses are analyzed with the LFG grammar, the parser will construct representations of complete parses, i.e. of all possible analyses that are compatible with the given lexicon and grammar. In the XLE system I used for my experiments, this representation is packed, i.e. common properties of several solutions are represented only once.

If we see as the goal of parsing to produce a unique solution, we can say that a packed representation is only a partial solution in the sense that it leaves some decisions open. A parser that has access only to the syntactical knowledge represented in the LFG descriptions adds too little information to the input string to allow for a unique result. Using notions from Information Theory, we can even quantify the amount of missing information, which is equal to the number of bits we would need to identify the right solution.

If we now use additional knowledge sources that give us preferences

for certain solutions, we will in general get a ranking of the analyses according to these preferences. How can we measure the quality of such a ranking?

An obvious measure is to always take the best solution according to this ranking (the Viterbi analysis), ignore everything else, and measure the fraction of the cases in which the parser did exactly right. This is essentially the measure called "labelled tree rate" in [Goodman1996]. A problem with this measure is that the precision measured in this way depends very strongly on the granularity in which the problem is given. Assume for an extreme example a very long and highly structured sentence[2]. The chance that a parser gets all details of such a sentence exactly right is practically 0, so the correctness of the Viterbi parse is not a very meaningful measure. Even if we measure the correctness on rather large corpora, the number of analyses that are exactly right in all details will be so small that their number is dominated by random effects and a meaningful comparison between different methods is not possible.

This fact has led researchers in the field to the definition of weaker notions of correctness, which give more meaningful results. One popular way of weakening the requirements is to look at certain properties of the result computed by the parser and determine the agreement between these properties and the manually selected reference solutions. For instance, we can calculate the number of constituents that have been correctly identified (i.e. agree in the category label, in start and end position) and calculate the ratio of correctly identified constituents to their overall number. A discussion of various evaluation mettrics is given in [Goodman1996], which also shows that a parsing algorithm that is trained in a way that maximizes the probability of being exactly right may achieve suboptimal results if measured with different criteria like precision and recall of the constituents or bracketings.

In our case, the goal is not only to identify a correct syntactic structure of the utterance, but also to get the functional structure right. We could decompose this goal into many simpler subgoals like getting the value of a specific feature correct and define evaluation criteria linked

[2]There are certain domains like patents or political texts where sentences with thousands of words are not unusual.

to such properties of the solutions. But the figures we would get in such an evaluation would depend strongly on the granularity of the properties we use, and more generally on the details of the LFG analyses we are assuming.

There is, however, a quite different way to achieve a weaker or finer grained evaluation metric. Instead of decomposing the analyses into simpler properties, we can stick to the goal of getting everything exactly right, but instead of using the Viterbi analysis, we accept the fact that the parser produces a ranked set of possible solutions. One way of seeing this situation is to assume that the result of parsing is a probability distribution over possible analyses. The goal is then to maximize the probability of the correct analysis under this distribution. In the ideal case, where the parser knows how to find the right result (and it knows that it knows this), this solution should get probability 1. In the rather unfortunate case where the parser has n possible results $s_1, \ldots, s_n$ and absolutely no way to distinguish between them, the best strategy would then be to assign a uniform probability of $1/n$ to each of them. If we randomly choose one of the solutions according to this uniform distribution, there is a chance of $1/n$ of making the right choice.

In the more interesting cases where we can exploit weak knowledge sources K to distinguish between the results, all available knowledge can be used to get a distribution that models the remaining uncertainty as accurately as possible. We would hence try to set up a distribution $p(s_i|w, K)$ in the hope that guessing the right solution according to this distribution will increase the chance of making the right choice. But how should we quantify the increase in parsing accuracy that a good ranking of the solutions brings about?

For the purposes of this chapter, I will apply a very simple and general method, in which the quality of a parser that produces ambiguous solutions annotated with probabilities is calculated as the chance of getting perfect results if the probabilities estimated by the parser are taken as the basis of a random choice. So if the parser is confronted with the sentences $\{w_1, \ldots, w_t\}$ and produces the solutions $\{\{\langle s_{1,1}, p_{1,1}\rangle, \ldots, \langle s_{1,n_1}, p_{1,n_1}\rangle\}, \ldots, \{\langle s_{t,1}, p_{t,1}\rangle, \ldots, \langle s_{t,n_t}, p_{t,n_t}\rangle\}\}$ and the correct analysis of sentence w_i is s_{c_i}, then the probability of an overall correct solution is given as $P_{corr} = \prod_i p_{i,c_i}$.

For long test sets, this probability will be very low, so we will mea-

sure this quantity on a scale that makes the numbers more manageable by taking the inverse and normalizing the number by taking the kth root, where k is some indicator of the size of the test set, such as the number of words, the number of utterances, or the the number of ambiguous utterances

This way of evaluating performance is essentially equivalent to standard measure of perplexity ([Jelinek1998]) of the complexity of a language source in work on speech recognition, hence we will call this number the *perplexity* of the disambiguation task. In the case of complete ignorance, where the same probability estimated gets assigned to every solution, this will lead to a value we call *baseline perplexity*

$$PP_{base} = \prod_i n_i^{1/k}$$

It can be interpreted as saying that guessing the right solution overall is as difficult as guessing, k times in sequence, the right value out of a hypothetical list of PP_{base} possibilities.

If we now have ways to make an informed probability estimate on the possible solutions, this will hopefully decrease the perplexity of the remaining decision problem. In the extreme case, where no uncertainty remains, the perplexity will be 1. The ratio between the baseline perplexity and the perplexity for the stochastic model will tell us how much of the original uncertainty could be resolved by stochastic disambiguation.

However, informal experiments show that the probability estimates returned by the stochastic model are qualitatively good, in the sense that they often give preference to the right analysis, but that the ratio between the estimated probabilities are too noisy to use the estimates directly. For instance, we can have the situation that the correct solution is always assigned to one of the first two ranks, but in some cases where it gets rank 2 the solution on rank 1 is estimated to be much more likely.

In such situations, there are a couple of strategies we can follow to get better estimates. It might not be too difficult to identify and inspect some of the most problematic cases and find an correction of the stochastic model that will improve them.

In practice, it often turns out that the disambiguation model is too

focussed, i.e. that it puts too much probability onto the solutions it prefers and leaves too little probability for the rest. This situation is quite frequent, and there are at least two different factors that contribute to this effect.

Typically, some intermediate probabilities are estimated in a too crude way, especially due to an unfortunate handling of sparseness problems. The estimation of unseen events gives the strongest fluctuations in the overall results.

Furthermore, the probabilistic model usually incorporates strong independence assumptions, as discussed in the last chapter. For instance, it might assume that the subject and the object of some verb are chosen independently. In reality, natural language phenomena are rarely independent. Hence the choice of a rare subject and a rare object at the same time might be much more probable than predicted by the model.

Both of these problems have already been discussed, and the Chapters 5, 6, and 7 provide techniques to improve the probability estimates in the face of these situations. However, the solutions proposed in these chapters are somewhat complex, and it is not immediately obvious how to combine them.

Instead of looking for fine-grained improvements to the individual parts of the probability model, we can also try to iron these problems problems out in a more general fashion and employ a mechanism to "translate" the original probability estimates of our model into a corrected estimate. There are many ways to do this, and we can use some of the techniques discussed in Section 5.4 to obtain a method for deriving correction formulas.

Before going into details, we have to decide what information we want to use for this purpose. One possibility is to ignore the actual probability estimates and use only the ranking information they provide. We could set up families of parametric distributions that map ranks into probabilities and use manually disambiguated training examples to select families of distributions and optimize the parameters. Instead of using only the rank information, we can additionally extract certain statistical properties of the estimated distributions, such as the entropy, and let these influence the corrected distribution.

Another possibility is to build a linear combination of the original estimates with a uniform distribution, where the mixing factors might

or might not depend on certain statistical properties of the original distribution. Mixing with a uniform distribution would essentially level out the tails of the original distribution, without affecting the high-probability estimates too much.

A third possibility is to reduce the slope of the distributions on a logarithmic scale in a uniform way, by raising the probabilities to some small power and renormalizing, i.e. setting $p'(s)$ to $p(s)^k / \sum_s p(s)^k$ with $0 < k < 1$

One can use training schemes for mixture distributions based on the EM algorithm to find optimal parameters for these possibilities, or even an optimal combination of several of these. However, this method should not be exaggerated, since optimizing a very detailed scheme for re-interpreting the original probabilities, based on a larger number of parameters, is likely to lead to an overlearning effect. On the other hand, the impact of this kind of overlearning is limited by the fact that all the transformations proposed above are monotonic, i.e. they do not influence the relative order of the solutions compared to the original ranking.

One can say that such a training procedure in a certain sense tries to identify meta-knowledge about the reliability of the original probability estimates.

In the sequel, I will not try to perform a sophisticated optimization, but use only a simple and regular special case. I will assume that we can abstract from the probability estimates and use only the ranking information, and I will additionally postulate that the corrected estimate is of the form

$$p'(rank = r | outcomes = n) = b^r / (\sum_{i=1}^{n} b^r)$$

This implies that there is a constant factor b by which a solution on rank $r + 1$ is less probable than the solution on rank r, and that this factor is constant not only across the solutions of one disambiguation problem, but globally[3].

[3]This assumption may not be justified in the presence of massive ambiguity, in which typicall a small set of high-quality solutions co-exists with a huge set of low-quality solutions. In such cases, it is more plausible to assume a Zipfian function to get from ranks back to probabilities.

Given some training data $\{w_1, \ldots, w_t\}$ that has been ranked by our model, such that input w_i has n_i solutions and the correct one is on rank r_i, we can ask ourselves which value of b will give the best prediction of the data, i.e. what value of b will maximize

$$p(d|M, b) = \prod_i \frac{b^{r_i} * (1-b)}{b(1-b^{n_i})}$$

In the cases where our ranking is very good, b should be small in order to give a relatively high weight to the solutions preferred by the model. In the cases where our ranking is almost random, a value of $b \approx 1$ will lead to an almost uniform distribution of the corrected probability estimates, i.e. the ranking produced by the model will be essentially ignored.

Unfortunately, there does not seem to be a simple way of computing the optimal value for b in the general case. We can simplify the situation by assuming that the number of outcomes per experiment is large compared to the rank of the correct solution, so that $1 - b^{n_i}$ can be approximated by 1. Then the value for b that maximizes the overall probability is given by

$$\hat{b} = arg\max_b \prod_i b^{r_1 - 1}(1-b) = arg\max_b b^{(\sum_i r_i) - t}(1-b)^t = 1 - \frac{t}{\sum_i r_i}$$

If we compare two models M' and M'', and assume that both of them can be approximated using the same value for b, the likelihood ratio depends only on the sum of the ranks these models give to the correct solutions

$$p(d|M', b)/p(d|M'', b) = b^{(\sum_i r'_i - \sum_i r''_i)}$$

I will take this (relatively informal) argumentation as a justification to use the sum of the solution ranks as a simple and meaningful indicator of the quality of a stochastic disambiguation model. To make the results less sensitive to the fraction of ambiguous and unambiguous instance, I will in the sequel use the *cumulated offset* $off = \sum_i (r_i - 1)$ as an indicator of the ranking quality, i.e. count the number of bad re-

sults one has to skip until one gets to the good ones[4]. There are cases in which the stochastic model cannot distinguish between competing analyses, because they differ only in the values of features for which no probabilities have been estimated. In this case, I use the arithmetic mean of the best and the worst possible rank the selected analysis could get if such ties are broken arbitrarily.

8.3 The Experiment

For the experiment, I use a large-scale LFG description of the English language, as it has been constructed by the NLTT group at Xerox Parc[Butt et al.1999] in the course of the ParGram project. This grammar is intended as a general-purpose resource, with (few) specific additions for peculiarities of the current domain.

The NLTT group took approximately 1000 sentences of technical text[5], from the user manual of HomeCentre, an integrated scanning/printing device produced by the Xerox corporation, sent it through the XLE parser, and manually disambiguated all the f-structure analyses. The result of this work is a bank of parsing charts in a compact representation, as a set of contexted facts, where the contexts are propositional clauses along the lines given in Chapter 2.

The items in this collection represent all possible pairs of syntax trees and functional structures (including their correspondences on subordinate levels) that are possible under the given LFG, plus a identification of the pairs that have been manually selected as the appropriate one in this context.

For the disambiguation, I used statistical data on the co-occurrences of English lemmata in certain linguistic relations. The data was obtained by applying IFSP (a shallow, finite-state-based parser, described in [Ait-Mokhtar and Chanod1997]) to the British National Corpus.

For the experiment, I implemented tools to read in from files the packed representations of f-structure charts, as they are produced by the

[4]This figure of merit has also a rather practical interpretation in the context of a system in which further processing of an unwanted analysis has a certain cost in terms of manual or computational effort.

[5]After removing duplicates and inconsistencies, 985 sentences remained

XLE implementation of the LFG formalism, to enumerate all possible readings that are represented in the chart, to extract from f-structures a set of predicates that appear in them, where each predicate is annotated with some simple characteristics of the context in which this predicate appears, and finally to determine the average rank of the solutions given in each of the sets under the given stoachastic model.

When extracting predicate information from a given f-structure, we record for all the PRED-features that appear in the structure one or several contextual stimuli, such as predicates of superordinate f-structures (or a dedicated value ROOT for PREDs on the outermost levels. There is a choice in the accuracy to which this context is taken into account. We can try to abstract away some details concerning the relation in which the context stands to the current PRED-value. In the most extreme case, we will conflate all possible relations and only keep the pure lexical information. This will obviously introduce inaccuracies, but on the other hand, it can reduce the number of different contexts to consider and hence slightly reduce the sparseness problem. Ideally, the prediction of a PRED can use both the full context and a simplified version of it, and use a suitable combination of these predictors, which can be optimized automatically.

This approach brings up the question what to do with multiple contextual evidence. In a sentence like *"The Status property page lets you monitor the progress of the current print job."*, *"you"* is at the same time object of *"lets"* and subject of *"monitor"*, which itself is embedded as an xcomp in the outermost structure. We can assume that the object is generated given both verbs, and in this case, we need a way to estimate probabilities for the object pred given both words in the context. One possibility for this would be to estimate trigram probabilities, but then the sparseness problem would become even more severe, especially as this type of construction is not that frequent in the available data.

For this feasibility study, I decided to sacrifice some accuracy and theoretical soundness, and use a simpler approximation. In cases where we have to use $p(x|y \wedge z)$ from estimators for $p(x|y)$ and $p(x|z)$, I will instead use a pseudo-probability

$$p'(x|y \wedge z) = max(p(x|y), p(x|z))$$

Correct results gets rank 1	224	46%
Correct results in top group	93	19%
Other	172	35%

Figure 8.1: Overall result of LFG disambiguation

If for any x, $p(x|y) \neq p(x|z)$, the result of such a point-wise maximization is not a probability distribution, as $\sum_x p'(x|y \wedge z) > 1$. However, given the difficulties to obtain reliable lexical probabilities in the first place, I am convinced that such a small inaccuracy will not have a significant impact on the quality of the results.

Given the rather limited amount of training data, it is important to exploit it to the maximal possible amount. But of course we need to evaluate the quality of the method on independent test data that did not contribute to the estimation of the parameters.

For the experiments, I used a leaving-one-out procedure in which only training instances different from the current annotated sentence are used to obtain the disambiguation parameters. However, it is not the current sentence itself, but only its annotation that should be kept out of the training. It is therefore legal to enumerate all possible analyses of the current sentence and to use fractional counts for the events that appear in these hypotheses. This can be seen as an application of the EM algorithm, but in the experiment, I did not try to find different probabilities for the possible readings before counting the fractional events.

The overall results of the experiment are given in Fig. 8.1. The percentages are based on the 489 cases that are actually ambiguous. These have, overall, 3203 analyses, which means that the average ambiguity *per ambiguous sentence* is 6.55.

In many cases, there was not a unique best result, as the ranking is based on the predicate-argument structure which does not reflect many of the finer distinctions made by the grammar. If we collapse the cases in which the correct result has the best rank or is in the top group, we can say that there is a success rate of 65%.

A slightly finer grained presentation of the results is given in Fig. 8.2. Here, the sentences have been grouped into three classes, according to the level of ambiguity they contain. To simplify the presentation, the

2-way ambiguity (239 charts):

Rank:	1	1-2	2
Cases:	140	60	39

3- or 4-way ambiguity (95 charts):

Rank:	1	x-2	x-3	x-4
Cases:	41	20	15	19

5- or more way ambiguity (155 charts):

Rank:	1	x-2	x-3	x-4	other
Cases:	43	20	16	20	56

Figure 8.2: Results grouped by amount of ambiguity

outcomes have been further clustered. A class of the form x-2 means that the correct result has been given rank 1-2 or 2, and analogous for the classes x-3 and x-4.

These results show that the method does fairly well for simple cases, but in the case of broader ambiguity, the ranks assigned by the method look somewhat unsatisfactory. It should be kept in mind that in the presense of broad ambiguity even one wrong decision made by the disambiguator can lead to the assignment of a very bad rank to the correct solution.

8.4 Future Work

There are several obvious ways in which the results reported in the last section could be improved.

Using more features: The current model uses only values of the PRED-feature and the syntactic functions to distinguish between the alternatives. It is a straightforward completion of this model to estimate and use probabilities for the remaining features. The fact that in less than half of the cases the correct analyses is unambiguously

given the highest rank is a strong motivation to explore the use of such additional knowledge sources. However, as there are no large corpora analyzed with the LFG grammar, the sentences in the treebank are essentially all data we can use to train this part of the model, which may lead to strong overlearning effects.

Smoothing against background models: The sparseness problem is a severe issue for all lexicalized models, so the use of "external" knowledge sources like lexical co-occurrences in large, general corpora is a promising way to improve the estimates. I have started experiments to automatically optimize mixtures of domain-dependent and external distributions in a way that is based on the techniques from Chapters 5 and 6. Unfortunately, this part of the experiment has not yet been completed and I have not yet obtained a ranking based on these mixtures.

Using "optimality marks": The LFG grammars make use of annotations for preferred and dispreferred syntactic constructions, see [Frank et al.1998] for a description of the formalism and its use. For any given input, this system singles out a set of so called optimal solutions, that satisfy the most important preferences and do not violate the most important dispreferences. Whereas the idea behind the system is that the correct solution should always be in the set of optimal solutions, it is not trivial to define the ranking between the marks in a way that actually guarantees this property. It is conceivable that a re-interpretation of the optimality marks as *soft, weighted preferences* could enable an automatic training procedure to induce their relative importance. This would lead to a joint ranking of optimal and suboptimal solutions, in which the optimality marks attached to the solutions contribute as one of several knowledge sources.

Incorporating more general dependencies: Even if compromises are made in some details, the stochastic model of f-structures used in the experiment follows basically the idea of a stochastic process that generates a structure piece by piece, where the probability of each decision in this process is estimated based on the material that is already present. The models that are introduced in Chapter 7 or described in [Abney1997, Riezler1998] do not follow this pattern. Here, arbitrary properties of a configuration can contribute to its probability, and the properties can refer to overlapping parts of the configurations, can have

cycles etc. This makes the models more expressive, but also the training of the weights considerably more involved. In connection with such models, it would be interesting to investigate whether for a small set of additional properties, added manually after an error analysis has revealed systematic problems in the stochastic disambiguator, weights could be trained automatically, and whether this can significantly improve the performance on unseen data.

Chapter 9

Conclusion and Outlook

9.1 Summary

This thesis has investigated two complementary approaches to cope with the ambiguity problem. In the first part, techniques for the compact encoding and efficient processing of (possibly large) sets of solutions have been reviewed. It has been shown that even exponential ambiguity can be handled in polynomial time by suitable algorithms that avoid unneccessary enumerations.

In the second part, several techniques for the estimation of linguistic probabilities and for the construction of stochastic language models have been proposed. The major problem for this type of models, the estimation of probabilities for unseen events, has been investigated in depth, and several novel techniques are given, that may contribute to the creation of stochastic language models of a new type. Finally, some of these techniques that had been introduced and studied in isolation, have been evaluated in the somewhat more ambitious task of ranking the results of ambiguous real-world utterances that have been processed with a large-scale LFG.

9.2 Future Work

Processing packed representations

Although some of the techniques discussed in the first part of this thesis are currently being used in practical implementations and allow to deal with large-scale grammars with satisfactory performance, there are several dimensions along which work should be done.

It would be useful to perform a more fine-grained analysis of the techniques described in [Maxwell and Kaplan1996b] and those in [Dymetman1997]. Whereas the latter is intended as a logical reconstruction of the techniques described in the former paper and actually implemented in the XLE system [Maxwell and Kaplan1996a], there are some differences between the approaches that need to be studied in more detail. For instance, the method given in [Dymetman1997] does not make use of a component that collects and processes propositional constraints on disjunctive "contexts" that are a crucial feature of the XLE implementation [Maxwell and Kaplan1989].

Another possible line of work would investigate how the techniques to process compact representations of sets of solutions can be used in applicatioons different from parsing, such as transfer or generation. Work in these areas has already lead to significant successes [Shemtov1997, Dymetman and Tendeau1998], but it would be interesting to investigate whether these problems can be seen as specific instances of a more general task of "transduction of ambiguous linguistic structures", whether general transduction algorithms can be found that avoid the enumeration of individual solutions, and whether efficient parsers, generators, transfer components, knowledge extractors and so on can be derived as special cases of such a schematic algorithm.

Estimating linguistic probabilities

The various techniques for estimation of linguistic probabilities that have been presented and discussed in this thesis can not more than scratch the surface of what needs to be done. The methods for smoothing lexical probability estimates that have been presented here and elsewhere need to be generalized so that a much richer context can

be taken into account. It seems plausible that log-linear or maximum entropy models can be used to achieve a theoretically sound combination of contextual evidence that does not lead to an explosion of the parameters that have to be estimated. The techniques for parameter estimation for maximum entropy models have to be improved to reduce their computational complexity, so that an application to linguistically rich grammatical resources that operate on large lexical domain becomes feasible. In the meantime, the techniques presented here might be generalized to operate as well on structurally richer models. Nothing in the way bigram probabilities are estimated in Section 6.3 is inherently limited to the two-dimensional case. Also, the generalization of similarity-based smoothing techniques to the multi-dimensional case should be relatively straightforward, once the computatitonal problems for the two-dimensional case are solved.

I am convinced that the combination of linguistic and stochastic knowledge, of hard rules and soft preferences, can lead to NLP systems of so far unattained quality and scope. However, the stochastic part of these models has been introduced for two quite different reasons. One one hand, statistical models are supposed to formalize linguistic preferences that cannot be expressed by hard rules. I believe that this part will remain, once the details have been worked out more precisely, as such models may be the most appropriate description of the linguistic reality. On the other hand, statistical models are used as an approximative replacement for knowledge that could be described symbolically, but has not been acquired. Statistical models can help to cope with missing lexical coverage, and can "simulate" semantic and other high-level knowledge, that so far has resisted formalization and/or capture.

This motivation for the use of stochatic models is as essential as the formalization of preferences, as we will never reach a state where all necessary knowledge will be completely available in a symbolic formalization.

However, these applications of stochastic models may have a more transient nature, as they allow the construction of processing techniques that are at the same time more accurate and more robust. Together with the steady improvement in computational resources and the availability of unmeasurable amounts of textual data, the hope is that these models will allow to construct large text corpora with automatically

high-quality annotations, from which missing knowledge can be extracted. Hence, the focus of the application of stochastic techniques as a replacement for missing knowledge will gradually shift away from lower levels like morphology and syntactic valency and start to adress even more difficult (and interesting) problems of semantics and discourse structure.

Integration of symbolic and stochastic processing

Another area where future work is needed is the integration of symbolic and stochastic processing techniques. In this thesis, both components where either juxtaposed, in the sense that the stochastic model was applied after the symbolic processor had delivered its result, or they were integrated in a relatively straightforward way, e.g. in a morphological transducer that incidentally adds up some log probs while doing its usual work. However, a much finer integration might be achievable if the scores produced in the stochastic model are used to reduce the amount of work the symbolic components have to perform. Using clever thresholding, we could avoid computation that cannot (or is not likely to) lead to highly ranked results, or we could at least try to postpone such computaitons until their need becomes obvious. Work in this direction is already being performed (see [Charniak, Goldwater, and Johnson1998, Goodman1998], to name only two), but a very interesting new direction would try to integrate stochastic preferences into transformation techniques that try to achieve interaction-free representations of sets of solutions, so that the algorithm concentrates on those parts of the representation that are likely to contribute to highly-ranked solutions.

Bibliography

[Abney1997] Abney, Steven. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

[Abney and Jonson1991] Abney, Steven and Mark Jonson. 1991. Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20(3):233–250.

[Ackley and Hinton1985] Ackley, D.H. and G.E. Hinton. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169.

[Agresti1990] Agresti, Alan. 1990. *Categorical Data Analysis*. John Wiley & Sons, New York.

[Aho and Ullman1972] Aho, Alfred V. and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*. Prentice Hall, Englewood Cliffs, N.J.

[Ait-Mokhtar and Chanod1997] Ait-Mokhtar, Salah and Jean-Pierre Chanod. 1997. Incremental finite-state parsing. In *ANLP'97*, pages 72–79, Washington.

[Alshawi1992] Alshawi, Hiyan, editor. 1992. *The Core Language Engine*. MIT Press, Cambridge, Massachusetts.

[Aït-Kaci1984] Aït-Kaci, Hassan. 1984. *A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures*. Ph.D. thesis, University of Pennsylvania, Philadelphia, Pa.

[Baker1979] Baker, J. 1979. Trainable grammars for speech recognition. In D. Klatt and J. Wolf, editors, *Speech Communication Papers*

for the 97th Meeting of the Acoustic Society of America. ASA, pages 547–50.

[Bar-Hillel1960] Bar-Hillel, Y. 1960. Automatic translation of languages. In Franz Alt, A. Donald Booth, and R.E. Meagher, editors, *Advances in Computers*. Academic Press, New York.

[Bar-Hillel, Perles, and Shamir1961] Bar-Hillel, Y., M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:142–172.

[Baum1971] Baum, L. E. 1971. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a markov process. In Shisha and Qved, editors, *Inequalities III*. Academic Press, New York, pages 1–8.

[Baum and Sell1968] Baum, Leonard E. and George R. Sell. 1968. Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227.

[Berger, Della Pietra, and Della Pietra1996]
Berger, Adam L., Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

[Billot and Lang1989] Billot, S. and B. Lang. 1989. The structure of shared forests in ambiguous parsing. In 27^{th} *Meeting of the Association for Computational Linguistics*.

[Black and othersl1991] Black, Ezra and othersl. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings, DARPA Speech and Natural Language Workshop*. pages 306–311.

[Block and Schmid1992] Block, H. Ulrich and Ludwig A. Schmid. 1992. Using disjunctive constraints in a bottom-up parser. In *Proceedings of KONVENS'92*, pages 169–177.

[Booth and Thompson1973] Booth, Taylor L. and Richard A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450.

[Brew1995] Brew, Chris. 1995. Stochastic HPSG. In *EACL 7*, pages 83–89.

[Brown et al.1992] Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40.

[Brown et al.1993] Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

[Brown et al.1992] Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

[Butt et al.1999] Butt, M., T.H. King, M.-E. Niño, and F. Segond. 1999. *A Grammar-Writer's Cookbook*. CSLI Publications, Stanford, CA.

[Carroll1996] Carroll, Glenn. 1996. Learning probabilistic grammars for language modelling. Technical report, Brown Computer Science Dept.

[Carroll and Charniak1992] Carroll, Glenn and Eugene Charniak. 1992. Learning probabilistic dependency grammars from labelled text. In *Proceedings of the AAAI Fall Symposium*, pages 25–32.

[Carroll and Rooth1998] Carroll, Glenn and Mats Rooth. 1998. Valence induction with a head-lexicalized pcfg. In *Third Conference on Empirical Methods in Natural Language Processing*.

[Charniak, Goldwater, and Johnson1998] Charniak, E., S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Workshop on Very Large Corpora*.

[Charniak1993] Charniak, Eugene. 1993. *Statistical language learning*. MIT Press, Cambridge, Mass.

[Chelba and Jelinek1998] Chelba, Ciprian and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Coling-ACL*.

[Chomsky1957] Chomsky, Noam. 1957. *Syntactic Structures*. Mouton, The Hague.

[Chomsky and Miller1963] Chomsky, Noam and George A. Miller. 1963. Finitary models of language users. In R. Luce, editor, *Handbook of Mathematical Psychology, volume 2*. John Wiley and Sons, New York, pages 419–491.

[Church and Gale1991] Church, Kenneth W. and William A. Gale. 1991. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computers, Speech, and Language*, 5.

[Collins1996] Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL-96*, Sant Cruz, CA, USA.

[Collins1997] Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL/EACL-97*, Madrid, Spain.

[Condon1928] Condon, E.V. 1928. Statistics of vocabulary. *Science*, 67.

[Corman, Leiserson, and Rivest1990] Corman, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts.

[Cover and Thomas1991] Cover, Thomas M. and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley and Sons, Inc., New York.

[Dagan, Pereira, and Lee1994] Dagan, Ido, Fernando C. N. Pereira, and Lillian Lee. 1994. Similarity-based estimation of word cooccurrence probabilities. In *ACL*.

[Darroch and Ratcliff1972] Darroch, J. N. and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43:1470–1480.

[Deerwester et al.1990] Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[Della Pietra, Della Pietra, and Lafferty1995]
Della Pietra, S., V. Della Pietra, and J. Lafferty. 1995. Inducing features of random fields. Technical report, CMU. cmp-lg/9506014.

[Dempster, Laird, and Rubin1977] Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistics Society, Series B*, 39:1–38.

[Dörre1997] Dörre, Jochen. 1997. Efficient construction of underspecified semantics under massive ambiguity. In *Proc. ACL*, Madrid.

[Dörre and Dorna1993] Dörre, Jochen and Michael Dorna. 1993. CUF - a formalism for linguistic knowledge representation. Deliverable R.1.2A, DYANA 2, August.

[Dörre and Eisele1989] Dörre, Jochen and Andreas Eisele. 1989. Determining consistency of feature terms with distributed disjunctions. In Dieter Metzing, editor, *GWAI-89. 13th German Workshop on Artificial Intelligence*, pages 270–279. Springer-Verlag. Informatik Fachberichte 216.

[Dörre and Eisele1990] Dörre, Jochen and Andreas Eisele. 1990. Feature logic with disjunctive unification. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki.

[Duda and Hart1973] Duda, Richard O. and Peter E. Hart. 1973. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York.

[Dymetman1997] Dymetman, Marc. 1997. Interaction-free grammars, chart-parsing, and the compact representation of ambiguity. In *Proc. IJCAI*, Nagoya.

[Dymetman and Tendeau1998] Dymetman, Marc and Frédéric Tendeau. 1998. An algorithm for the transfer of packed linguistic structures. unpublished manuscript.

[Earley1970] Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

[Eisele1987] Eisele, Andreas. 1987. Eine Implementierung rekursiver Merkmalstrukturen mit disjunktiven Angaben. Diplomarbeit, Institut für Informatik, Universität Stuttgart, Stuttgart. in German.

[Eisele1994] Eisele, Andreas. 1994. Towards probabilistic extensions of constraint-based grammars. In Jochen Dörre, editor, *Computational Aspects of Constraint-Based Linguistic Description II, DYANA-2 deliverable R1.2.B*. ESPRIT, Basic Research Project 6852, September.

[Eisele and Dörre1990] Eisele, Andreas and Jochen Dörre. 1990. Disjunctive Unification. IWBS Report 124, IWBS, IBM Deutschland, Stuttgart, May.

[Eisner1996] Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*, Kopenhagen, Denmark.

[Essen and Steinbiss1992] Essen, Ute and Volker Steinbiss. 1992. Coocurrence smoothing for stochastic language modeling. In *ICASSP*, volume I. pages 161–164.

[Estoup1916] Estoup, J.B. 1916. *Gammes Sténographiques*. Paris.

[Frank et al.1998] Frank, Anette, Tracy Holloway King, Jonas Kuhn, and John Maxwell. 1998. Optimality theory style constraint ranking

in large-scale LFG grammars. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG98 Conference, University of Queensland, Brisbane*. CSLI Online Publications, Stanford, CA.

[Franz1997] Franz, Alexander. 1997. Independence assumptions considered harmful. In *Proceedings of ACL 35 and EACL 8*, pages 182–189, Morristown NJ. Association of Computational Linguistics.

[Fujii1998] Fujii, Atsushi. 1998. Corpus-based word sense disambiguation. Technical report, Tokyo Institute of Technology.

[Geman and Geman1985] Geman, Stuart and Donald Geman. 1985. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 6(6):721–42.

[Good1953] Good, I. J. 1953. On the population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

[Good1956] Good, I. J. 1956. On the estimation of small frequencies in contingency tables. *J. Roy. Statisti. Soc., B*, 18:113–124.

[Good1964] Good, I. J. 1964. *The estimation of probabilities*. MIT Press, Cambridge, MA.

[Goodman1996] Goodman, Joshua. 1996. Parsing algorithms and metrics. In *Proceedings of ACL-96*, Sant Cruz, CA, USA.

[Goodman1998] Goodman, Joshua. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University, Cambridge, MA.

[Graham, Harrison, and Ruzzo1980] Graham, S.L., M.A. Harrison, and W.L. Ruzzo. 1980. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems*, 2(3):415–462.

[Grefenstette1994] Grefenstette, G. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Press, Boston.

[Hasida1986a] Hasida, Kôiti. 1986a. Conditioned unification fro natural language processing. In *Coling*, pages 85–87.

[Hasida1986b] Hasida, Kôiti. 1986b. Sentence processing as constraint transformation. In *ECAI*, pages 339–344.

[Höhfeld and Smolka1988] Höhfeld, Markus and Gert Smolka. 1988. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, W. Germany, October. To appear in the Journal of Logic Programming.

[Hindle and Rooth1991] Hindle, Donald and Mats Rooth. 1991. Structural ambiguity and lexical relations. In *Proceedings of ACL*, pages 229–236.

[Hindle and Rooth1993] Hindle, Donald and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 18:103–120.

[Hofmann and Puzicha1998] Hofmann, Thomas and Jan Puzicha. 1998. Statistical models for co-occurrence data. Technical report, Massachusetts Institute of Technology.

[Ide and Véronis1998] Ide, Nancy and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40.

[Jaynes1957] Jaynes, Edwin T. 1957. Information theory and statistical mechanics. *Physical Review*, 106:620–630.

[Jaynes1995] Jaynes, Edwin T. 1995. *Probability Theory: The Logic of Science*. Fragmentary edition. available from http://omega.math.albany.edu:8008/JaynesBook.html.

[Jelinek1976] Jelinek, F. 1976. Continuous speech recognition by statistical methods. In *Proceedings of the IEEE*, volume 64, pages 532–56.

[Jelinek1998] Jelinek, Frederick. 1998. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge MA.

[Jelinek and Mercer1980] Jelinek, Frederick and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands. North-Holland.

[Jelinek, Mercer, and Roukos1992] Jelinek, Frederick, Robert L. Mercer, and Salim Roukos. 1992. Principles of lexical language modeling for speech recognition. In Sadaoki Furui and M. Mohan Sondhi, editors, *Advances in Speech Signal Processing*. Mercer Dekker, Inc.

[Johnson1996] Johnson, Mark. 1996. Ranking LFG analyses. unpublished manuscript.

[Johnson and Dörre1995] Johnson, Mark and Jochen Dörre. 1995. Memoization of coroutined constraints. In *ACL95*, pages 100–107, San Francisco. Morgan Kaufmann.

[Johnson et al.1999] Johnson, Mark, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of ACL*.

[Juang and Rabiner1992] Juang, Biing-Hwang and Lawrence R. Rabiner. 1992. Issues in using hidden markov models for speech recognition. In Sadaoki Furui and Mohan M. Sondhi, editors, *Advances in Speech Signal Processing*. Marcel Dekker, New York.

[Kanamori and Horiuchi1987] Kanamori, Tadashi and Kenji Horiuchi. 1987. Construction of logic programs based on generalized unfold/fold rules. In Jean-Louis Lassez, editor, *Logic Programming: Proceedings of the Fourth International Conference*, volume 2, pages 744–768, Cambridge, Massachusetts. The MIT Press.

[Kaplan and Bresnan1982] Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-Functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, chapter 4, pages 173–281.

[Kaplan and Kay1994] Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(331-378):1–40.

[Karttunen1984] Karttunen, Lauri. 1984. Features and values. In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 28–33, Stanford, Cal.

[Karttunen1994] Karttunen, Lauri. 1994. Constructing Lexical Transducers. In *Proceedings of the 15th International Conference on Computational Linguistics, Coling*, Kyoto, Japan.

[Kasper1987] Kasper, Robert T. 1987. A unification method for disjunctive feature descriptions. In *Proceedings of the 25th Annual Meeting of the ACL, Stanford University*, pages 235–242, Stanford, Cal.

[Kasper and Rounds1986] Kasper, Robert T. and William C. Rounds. 1986. A logical semantics for feature structures. In *Proceedings of the 24th Annual Meeting of the ACL, Columbia University*, pages 257–265, New York, N.Y.

[Katz1987] Katz, Slava M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401.

[Kay1979] Kay, Martin. 1979. Functional grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society*, Berkeley, CA.

[Kernighan, Church, and Gale1990] Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Coling*, pages 205–210.

[Knight1989] Knight, K. 1989. Unification: A multidisciplinary survey. *Association for Computing Machinery, Computing Surveys*, 21(1):93–124.

[Koskenniemi1983] Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. University of Helsinki.

[Krenn and Samuelsson1996] Krenn, Brigitte and Christer Samuelsson. 1996. The linguist's guide to statistics. Technical report, Universität des Saarlandes.

[Kuhn and De Mori1990] Kuhn, Roland and Renato De Mori. 1990. A cache-based natural language model for speech recognition.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(6):570–583.

[Lafferty, Sleator, and Temperley1992] Lafferty, John, Daniel Sleator, and Davy Temperley. 1992. Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the AAAI-92 Fall Symposium: Probabilistic Approaches to Natural Language*, pages 89–97.

[Lang1994] Lang, Bernard. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10.

[Lari and Young1990] Lari, K. and S.J. Young. 1990. The Estimation of Stochastic Context-Free Grammars Using the Inside-Outside Algorithm. *Computers, Speech, and Language*, 4(1):35–56.

[Lee1997] Lee, Lillian. 1997. Similarity-based approaches to natural language processing. Technical Report TR-11-97, Harvard University.

[Levinson1983] Levinson, Stephen C. 1983. *Pragmatics*. Cambridge University Press, Cambridge, England.

[Lyons1968] Lyons, John. 1968. *Introduction to theoretical linguistics*. Cambridge University Press, Cambridge, England.

[Lyons1977] Lyons, John. 1977. *Semantics*. Cambridge University Press, Cambridge, England.

[Mandelbrot1954] Mandelbrot, Benoit. 1954. Structure formelle des textes et communication. *Word*, 10(1–27).

[Manning and Schütze1998] Manning,
Christopher and Hinrich Schütze. 1998. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge MA. In progress.

[Mark et al.1992] Mark, Kevin, Michael Miller, Ulf Grenander, and Steve Abney. 1992. Parameter estimation for constrained context-free language models. In *Proceedings, Fifth DARPA Speech and Natural Language Workshop*. Morgan Kaufman, San Mateo, CA.

[Maxwell and Kaplan1989] Maxwell, John T. and Ronald M. Kaplan. 1989. An overview of disjunctive constraint satisfaction. In *Proceedings of the International Workshop on Parsing Technologies*, pages 18–27, Pittsburgh, PA.

[Maxwell and Kaplan1993] Maxwell, John T. and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590, December.

[Maxwell and Kaplan1996a] Maxwell, John T. and Ronald M. Kaplan. 1996a. An efficient parser for LFG. In *First LFG Conference*, Grenoble, France, August. `http://www-csli.stanford.edu/user/mutt/`.

[Maxwell and Kaplan1996b] Maxwell, John T. and Ronald M. Kaplan. 1996b. Unification-based parsers that automatically take advantage of context freeness. unpublished manuscript.

[McLachlan and Krishnan1998] McLachlan, G.J. and T. Krishnan. 1998. *The EM Algorithm and Extensions*. John Wiley & Sons, New York.

[Miller et al.1990] Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244.

[Nadas1984] Nadas, Arthur. 1984. Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-32(4):859–861.

[Nadas1985] Nadas, Arthur. 1985. On Turing's formula for word probabilities. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-33(6):1414–1416.

[Nakano1991] Nakano, Mikio. 1991. Constraint projection: An efficient treatment of disjunctive feature descriptions. In *Proceedings of the Twenty-Ninth Annual Meeting of the ACL*, pages 307–314, Berkeley, CA. Association for Computational Linguistics.

[Nelson and Oppen1980] Nelson, Greg and Derek C. Oppen. 1980. Fast decision procedures based on congruence closure. *Journal of the Association for Computing Machinery*, 27(2):356–364.

[Ney and Essen1993] Ney, Hermann and Ute Essen. 1993. Estimating 'small' probabilities by leaving-one-out. In *European Conference on Speech Communication and Technology*. Berlin, Germany, pages 2239–2242.

[Ney, Essen, and Kneser1994] Ney, Hermann, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilisitic dependences in stochastic language modeling. *Computers, Speech, and Language*, 8(1):1–38.

[Nießen et al.1998] Nießen, S., S. Vogel, H. Ney, and C. Tillmann. 1998. A DP based search algorithm for statistical machine translation. In *Coling-ACL*, pages 960–967.

[O'Boyle, Owens, and Smith1994] O'Boyle, P., M. Owens, and F. J. Smith. 1994. A weighted average n-gram model of natural language. *Computers, Speech, and Language*, 8(4).

[Oflazer1996] Oflazer, Kemal. 1996. Error-tolerant finite state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–90.

[Pearl1988] Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo.

[Pereira and Schabes1992] Pereira, Fernando C. N. and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*, pages 128–135.

[Pereira, Tishby, and Lee1993] Pereira, Fernando C. N., Naftali Z. Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *ACL*, pages 183–190.

[Pereira and Warren1983] Pereira, Fernando C. N. and David H. D. Warren. 1983. Parsing as deduction. In *Proceedings of the 21th*

Annual Meeting of the Association for Computational Linguistics, pages 137–144, MIT, Cambridge, MA, June.

[Pereira and Warren1980] Pereira, Fernando C.N. and David H.D. Warren. 1980. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278.

[Pereira and Warren1983] Pereira, Fernando C.N. and David H.D. Warren. 1983. Parsing as deduction. In *The Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144, M.I.T., Cambridge, Massachusetts.

[Pollard and Sag1987] Pollard, Carl and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics, Volume I.* Number 13 in CSLI Lecture Notes. CSLI, Stanford University.

[Riezler1998] Riezler, Stefan. 1998. Probabilistic constraint logic programming. Technical report, Universität Tübingen.

[Rissanen1989] Rissanen, Jorma. 1989. *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore.

[Romesburg1990] Romesburg, H. Charles. 1990. *Cluster Analysis for Researchers.* Krieger Publishing Company, Malabar, Florida.

[Rooth1995] Rooth, Mats. 1995. Two-dimensional clusters in grammatical relations. Paper presented at IJCAI Lexicon workshop.

[Samuelsson1996] Samuelsson, Christer. 1996. Relating turing's formula and zipf's law. *Procs. 4th Workshop on Very Large Corpora*, pages 70–78.

[Saul and Pereira1997] Saul, L. and F. Pereira. 1997. Aggregate and mixed-order markov models for statistical language processing. In *Second Conference on Empirical Methods in Natural Language Processing.*

[Schütze1992] Schütze, Hinrich. 1992. Dimensions of meaning. In *Proceedings of Supercomputing '92*, pages 787–796, Minneapolis MN.

[Schütze1993] Schütze, Hinrich. 1993. Word space. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann Publishers, San Mateo CA, pages 895–902.

[Schütze1997] Schütze, Hinrich. 1997. *Ambiguity Resolution in Language Learning*. Number 71 in CSLI Lecture Notes. CSLI Publications, Stanford, CA.

[Shannon1948] Shannon, Claude E. 1948. The mathematical theory of communication. *Bell System Technical Journal*, 27:398–403.

[Shemtov1997] Shemtov, Hadar. 1997. *Ambiguity Management in Natural Language Generation*. Ph.D. thesis, Stanford.

[Sleator and Temperley1991] Sleator, D. and D. Temperley. 1991. Parsing english with a link grammar. Technical report, Carnegie Mellon University.

[Smolka1988] Smolka, Gert. 1988. A feature logic with subsorts. LILOG Report 33, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, W. Germany, May.

[Smolka1993] Smolka, Gert. 1993. Residuation and guarded rules for constraint logic programming. In Frédéric Benhamou and Alain Colmerauer, editors, *Constraint Logic Programming: Selected Research*. The MIT Press, Cambridge, Mass, chapter 22, pages 405–419.

[Sánchez and Benedi1997] Sánchez, J-A and J-M Benedi. 1997. Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1052ff.

[Tamaki and Sato1984] Tamaki, Hisao and Taisuke Sato. 1984. Unfold/Fold transformation of logic programs. In S. Å. Tärnlund, editor, *Proceedings of the Second International Conference on Logic Programming*, pages 127–138, Uppsala, Sweden.

[Tsuda1993] Tsuda, Hiroshi. 1993. A guide to CU-prolog III. Technical report, ICOT.

[Tsuda, Hasida, and Sirai1989] Tsuda, Hiroshi, Koiti Hasida, and Hidetosi Sirai. 1989. JPSG parser on constraint logic programming. In *Proceedings of 4th ACL European Chapter*, pages 95–102.

[Uszkoreit1984] Uszkoreit, Hans Jürgen. 1984. *Word Order and Constituent Structure in German.* Ph.D. thesis, Univ. of Texas, Austin, December.

[van Noord1995] van Noord, Gertjan. 1995. The intersection of finite state automata and definite clause grammars. In *ACL95*, San Francisco. Morgan Kaufmann.

[Winkler1995] Winkler, Gerhard. 1995. *Image Analysis, Random Fields, and Dynamic Monte Carlo Methods.* Springer.

[Wu and Wong1998] Wu, Dekai and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Coling-ACL*, pages 1408–1415.

[Younger1967] Younger, D. H. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.