ENRICHING LINGUISTICS WITH STATISTICS:

PERFORMANCE MODELS OF NATURAL LANGUAGE

Rens Bod

ILLC Dissertation Series 1995-14

Rens Bod
Department of Computational Linguistics
Universiteit van Amsterdam
Spuistraat 134, 1012 VB Amsterdam
The Netherlands
phone: +31-20-5252079
fax: +31-20-5254429
e-mail: Rens.Bod@let.uva.nl

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-5256090
fax: +31-20-5255101
e-mail: illc@fwi.uva.nl

ENRICHING LINGUISTICS WITH STATISTICS:

PERFORMANCE MODELS OF NATURAL LANGUAGE

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
Prof. dr P.W.M. de Meijer
ten overstaan van een door het college van dekanen ingestelde
commissie in het openbaar te verdedigen in de Aula der Universiteit
(Oude Lutherse Kerk, ingang Singel 411, hoek Spui)

op woensdag 13 september 1995 te 12.00 uur
door Laurens Wilhelmus Maria Bod
geboren te Bergh

Faculteit der Letteren
Universiteit van Amsterdam
Promotor: Prof.dr.ir. R.J.H. Scha

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Bod, Rens

Enriching linguistics with statistics: performance models
of natural language / Rens Bod. - Amsterdam : Institute
for Logic, Language and Computation, Universiteit van
Amsterdam. - Ill. - (ILLC dissertation series ; 1995-14)
Proefschrift Universiteit van Amsterdam. - Met lit. opg.
ISBN 90-74795-36-6
NUGI 941
Trefw.: computerlinguïstiek

Printed in the Netherlands by Academische Pers, Amsterdam.

Acknowledgements

This thesis benefitted from discussions with many people. I would like to express my thanks to Martin van den Berg, Kenneth Church, Marc Dymetman, Bipin Indurkhya, Laszlo Kalman, Ronald Kaplan, Martin Kay, Steven Krauwer, Kwee Tjoe Liong, Neza van der Leeuw, David Magerman, Arie Mijnlieff, Fernando Pereira, Philip Resnik, Yves Schabes, Khalil Sima'an and Frederik Somsen. Furthermore, I wish to thank the members of the graduation committee: Renate Bartsch, Jan van Eijck, Gerard Kempen, Chris Klaassen and Anton Nijholt. I am grateful to Steven Krauwer for allowing me to work at this thesis while I was involved in the CLASK project ("Combining Linguistic and Statistical Knowledge") at Utrecht University. The fruitful discussions and positive cooperation with my colleagues Martin van den Berg and Khalil Sima'an have been of incalculable value.

Special thanks go to my supervisor Remko Scha, who I could - and did - bother with my research any time I wanted, twenty-four hours a day, seven days a week.

I wish to thank my parents for supporting me for many years.

Parts of this thesis were previously published in (Bod, 1992a), (Bod, 1992b), (Bod, 1992c), (Bod, 1993a), (Bod, 1993b), (Bod, 1993c), (Bod, 1993d), (Bod, 1993e), (Bod, 1993f), (Bod, Dastani, Prust, Scha & Zeevat, 1993), (Bod & Scha, 1993), (Scha & Bod, 1993), (vdBerg, Bod & Scha, 1994), (Bod, Krauwer & Sima'an, 1994), (Bod & Scha, 1994), (Sima'an, Bod, Krauwer & Scha, 1994), (Bod, 1995).

Dedico questa tesi a Daniela e Livio

Contents

Chapter 1

Introduction: Enriching Linguistics with Statistics

One of the most fundamental concepts in current linguistic theory is the *competence-performance* dichotomy. A linguistic *competence* model aims to characterize the set of grammatical sentences together with the possible analyses that can be assigned to these sentences; a linguistic *performance* model aims to describe the actual production and perception of natural language sentences in specific situations (Chomsky, 1957, 1965). This dichotomy has become the methodological paradigm for all formal linguistic theories. It is assumed that the primary goal of linguistics is the development of a theory of language *competence*. Linguistic theory seems to have given up on the problem of language *performance*. It has adopted the formal languages of logic and mathematics as its paradigm examples: a language is viewed as a set of sentence/meaning pairs that is explicitly characterized by a complete and consistent system of recursive rules. No attempt is made to specify how such a competence model might be implemented or approximated by psychologically plausible processes of language perception and production.

We may illustrate this dichotomy by focussing on the problem of disambiguation. Existing linguistic theories account very well for the fact that most natural language sentences have an extremely large number of possible analyses (and corresponding semantic interpretations). But they do not account for the fact that human language comprehenders usually perceive only one or two of these analyses.[1] A performance theory of natural language, on the other hand, should model the input-output properties of actual human perception. It should not be satisfied with describing the space of possible analyses that input sentences may get; it should predict which analyses comprehenders actually assign to their input sentences.

It is the main goal of this dissertation to show how a statistical enrichment of a linguistic competence theory can model the input-output properties of human language perception. In doing so, we will concentrate on the problem of disambiguation. We will start by motivating a statistical approach to linguistics, after which we will deal with the question of what is involved in creating a statistical performance model which can select from all possible analyses of a sentence, the analysis which is actually perceived.

1.1 Motivations for a statistical approach to linguistics

A long series of psychological investigations indicate that (1) people register frequencies and differences in frequencies (e.g. Hasher and Chromiak, 1977; Kausler and Puckett, 1980; Pearlmutter & MacDonald, 1992), (2) people prefer

---

[1] The combinatorial explosion of syntactic analyses (and corresponding semantic interpretations) of natural language sentences is often underestimated in linguistics. In the field of psycholinguistics and language technology, it is a widely studied subject (e.g. Church & Patil, 1983; Frazier, 1979, 1987; MacDonald et al. 1994; Martin, Church & Patil, 1983; Tanenhaus et al., 1979). The following example from (Martin, Church & Patil, 1983), shows four sentences together with the number of different analyses. The ambiguity explosion due to different possible attachments of prepositional phrases and relative clauses is enormous; nevertheless these sentences do not appear to be difficult for people to understand.

List the sales of products in 1973	3
List the sales of products produced in 1973	10
List the sales of products in 1973 with the products in 1972	28
List the sales of products produced in 1973 with the products produced in 1972	455

analyses that have been experienced before: "old before new" (e.g. Hasher and Zacks, 1984; Jacoby & Brooks, 1984; Fenk-Oczlon, 1989), and (3) the preference for "old before new" is influenced by the frequency of occurrence of analyses: "more frequent before less frequent" (e.g. Fenk-Oczlon, 1989; Mitchell et al., 1992; Juliano & Tanenhaus, 1993). We can summarize the above statements by (4): *the frequencies of previously perceived analyses bias the analysis of a new input.*

These statements form an important motivation for a frequency-based approach to a model which aims to characterize human language performance. The use of probability theory seems a straightforward step, since it models the notion of *frequency of occurrence* in a mathematically precise way, offering a coherent and well understood framework. We therefore rewrite the statements (1) through (4) by (5): *a comprehender tends to perceive the most probable analysis of a new input on the basis of frequencies of previously perceived analyses*.

Note that (5) does not say that other possible analyses of a sentence are "wrong". An analysis is formally "correct" if it can be generated by the underlying linguistic theory (the competence model). It is the task of a performance model to select among all correct analyses the analysis which is actually *perceived* by a comprehender. This does not mean that a comprehender is unable to perceive the other analyses of an input. He/she has only a very strong tendency towards perceiving a more probable analysis with respect to his/her previous language experiences. What (5) does imply, is, that different linguistic experiences can yield different perceived analyses of sentences. It is interesting to note that there is some support for this implication (Mitchell, Cuetos & Corley, 1992). However, in the absence of actual collections of individual language experiences, we will abstract from these individual differences, and limit ourselves to actually available collections of analyzed natural language utterances (see section 1.3).

Besides from a psychological point of view, it is also possible to motivate a statistical approach from an engineering point of view. Statistical extensions of linguistic theories have recently gained a vast popularity in the field of natural language processing. In this field, it is widely recognized that purely rule-based methods suffer from lack of robustness in solving uncertainty due to overgeneration (if too many analyses are generated for a sentence) and undergeneration (if no analysis is generated for a sentence). A statistical approach is supposed to be more robust than a purely rule-based approach, since it allows for *a best guess in case of uncertainty* (cf. Garside et al., 1987; Liberman, 1991; Liberman & Schabes, 1993; Charniak, 1993). The succes of statistical methods in the field of speech recognition has further reinforced this insight (cf. Church & Mercer, 1993).

1.2 What should we demand from a statistical enrichment of a linguistic theory?

What does a statistical extension of a linguistic theory, often referred to as a "stochastic grammar", look like? In the literature, we can observe the following recurrent theme: (1) take your favorite linguistic theory (a competence model), (2) attach application probabilities to the productive units of this theory. Examples are stochastic context-free grammar (Suppes, 1970; Fujisaki, 1984; Sampson, 1986), stochastic tree-adjoining grammar (Resnik, 1992; Schabes, 1992), stochastic unification-based grammar (Briscoe & Carroll, 1993). It would be easy to define, in a similar way, stochastic lexical-functional grammar, stochastic categorial grammar, etcetera.

Though they use different underlying competence models, these stochastic grammars have in common that they do not fully satisfy statement (4) in section 1.1, saying that the frequencies of previously experienced analyses bias the analysis of a new input. What these stochastic grammars take into account are only the frequencies of the single linguistic units that make up analyses according to the underlying competence model. As a consequence, these stochastic grammars have only had a modest success in solving ambiguity in natural language.

This brings us to a more fundamental point. If application probabilities are assigned to the single productive units of an underlying competence model, it is tacitly assumed that the statistical units coincide with the linguistic units. In other words, it is assumed that there are no statistical dependencies that go beyond the linguistic dependencies described by the competence theory. This assumption is wrong. There can be many statistically interesting dependencies that do not coincide with the linguistically interesting dependencies according to the competence model (for examples, see chapter 3, section 4 and chapters 5, 6 and 7). Since we do not know beforehand which dependencies may be statistically interesting, we should not constrain or predefine the statistical units, but *take all, arbitrarily large (previously experienced) structures and interpretations as possible statistical units*. This will be the working hypothesis for this thesis.

1.3 The Data-Oriented Parsing framework: performance models of natural language

What we ideally need in order to derive the probability of an analysis of a sentence is a very large language corpus, that stands for a person's past language experience, in which each sentence is annotated with a (syntactic, semantic, pragmatic) analysis that seemed most appropriate for understanding the sentence in the context in which it was uttered. The probability of an analysis of an input sentence can then be derived by taking the relative frequency of this analysis in the corpus. But how can we derive the probability of an analysis of an input sentence which does not occur in the corpus? From our hypothesis that we should take any arbitrarily large structure or interpretation as a possible statistical unit, we propose that an analysis of a new sentence be constructed out of arbitrarily large *sub-analyses* that occur in the corpus. By combining the probabilities of these sub-analyses in a statistically adequate way, we can *estimate* the probability of a new analysis. In order to actually accomplish this, the following parameters need to be specified:

(1) definition of the sentence-analyses

(2) definition of the sub-analyses

(3) definition of the combination-operations between sub-analyses

(4) definition of the combination-probabilities between sub-analyses

The definition of the sentence-analyses follows from the competence model one wishes to use for annotating the corpus sentences; each sentence must be annotated with the analysis that is most appropriate for understanding the sentence in the context in which it was uttered (that is, the *perceived* analysis). The definition of the sub-analyses specifies the space of (arbitrarily large) statistical units which include the linguistic units. The combination-operations specify how sub-analyses can be combined into new analyses; these operations may be equal to the combination-operations of the competence model, though this need not necessarily be so. The combination-probabilities define how the probability of combining sub-analyses is calculated from the frequencies of occurrence of these sub-analyses in the corpus of sentence-analyses.

The above parameters constitute the prerequisites of what we will call a "performance model of natural language". This notion of performance model has become known under the name of "Data-Oriented Parsing framework" or "DOP framework", and was introduced in (Scha, 1990) and developed in (Bod, 1992a, 1993a). The predicate "Data-Oriented" refers to the use of actual corpus data, while the word "Parsing" indicates the prime interest in (syntactic) language perception. Note, however, that the DOP framework may as well be used for language *production*.

It is evident that the above parameters can be instantiated in many different ways, depending on the criteria used. The parameters indicate a framework by means of which a wide range of performance models can be defined. In this thesis, we will instantiate above parameters only in as far as the resulting performance models can be implemented and tested with actually available corpora. Unfortunately, current linguistic theories have not been very much concerned with creating a corpus of analyzed sentences. The only corpora that are available now consist of syntactically labeled phrase structure trees, such as the Nijmegen Corpus (van Halteren & Oostdijk, 1988), the Lancaster Treebank (Black, Garside & Leech, 1993) and the Pennsylvania Treebank (Marcus et al., 1993). The underlying competence models used for these sentence-analyses are thus very shallow, which means that a performance model based on these corpora can at best account for the syntactic dimension of language performance. Although the creation of richer analyzed language corpora is under development (see chapter 8), we will limit ourselves in this thesis to the corpora that are currently available.

We want, however, to stress that our notion of performance model does not stand in opposition to knowledge-based approaches to language processing. In the literature, statistical approaches are usually opposed to knowledge-based approaches (cf. Alshawi, 1994; Price, 1994). This is because statistical methods have often been claimed to substitute for knowledge-based methods. We believe that in a fully analyzed language corpus, the analyses must be enriched with a maximum of pragmatic and world-knowledge information. But we also believe that an adequate performance model should take the frequencies of these analyses into account.

1.4 Evaluation of performance models: linguistics as an experimental science

Before we go into the details of instantiations of performance models, we need to say a word about the problem of evaluation. A performance model of natural language may be evaluated in many ways: by testing the coverage of the model, by testing the timing of finding the most probable analysis, by testing the accuracy of the most probable analysis, etc. Since we focus on the problem of language disambiguation in this thesis, we will put emphasis on testing the adequacy of the input-output behavior of a performance model. This means that we will concentrate first of all on the analysis *results* of a performance model and not so much on the analysis *process*.

We believe that serious evaluations of analysis results of natural language systems are still being neglected. There is no consensus on the kind of test procedure, on the establishment of the appropriate or perceived analysis, on the way of comparing two analyses. As a starter, let us look at a test procedure in the field of natural language parsing. In this field, the following procedure has traditionally been used: (1) select a set of test sentences, (2) let the parsing system calculate the "best" analyses, and (3) let a linguist decide whether the "best" analyses are "appropriate". This procedure has at least one drawback: if a linguist decides whether an experimentally obtained analysis is "appropriate" (i.e. corresponding to the perceived analysis), he is extremely biased by this analysis. He may judge an analysis as appropriate, while he would have assigned a completely different analysis to this sentence if he had not been confronted by the experimentally generated analysis before. This phenomenon of being influenced by experimental outcomes is well known; nevertheless, this evaluation procedure is still being used by many parsing systems, which as a consequence report unreliably high system accuracies (e.g. Simmons & Yu, 1992: 99.7%; Karlsson et al., 1995: 99.9%).

In order to exclude the influence of experimental outcomes on the appropriateness judgments, we believe that it is of utmost importance to establish the appropriate analyses of the test sentences *beforehand* and not afterwards. Moreover, the appropriate analyses should preferably not be established by the experimenter himself, but by a separate (group of) annotator(s). A test procedure which incorporates these demands is known under the name of *blind testing method*. This method, which has been advocated by the ARPA community (cf. (D)ARPA Proceedings, 1989-1994), dictates that a manually analyzed language corpus is randomly divided into a training set and a test set, where the analyses from the test set are kept aside. The analyses from the training set may be used to train the system, while the sentences of the test set are parsed by the system. The degree to which the most probable analyses generated by the system match with the test set analyses is a measure for the accuracy of the system.

The question is now as to what kind of *accuracy metric* is most adequate to compare the most probable analyses generated by the system with the analyses in the test set. In the ARPA community, the most popular accuracy metric is the so-called *bracketing accuracy*, defined as the percentage of brackets of the analyses that are not "crossing" the bracketings in the test set analyses (e.g. Black et al., 1991; Harrison et al., 1991; Pereira & Schabes, 1992; Grishman et al., 1992; Schabes et al., 1993; Magerman, 1995). A reason for the popularity of this metric lies in the fact that it allows for the evaluation of systems using different grammar formalisms. We believe that the notion of bracketing accuracy is too poor for measuring the accuracy of a performance model. In testing a performance model, we are interested in whether the model is able to correctly select the perceived analysis from the possible analyses of a test sentence. Therefore we need a metric which demands an exact match between the most probable analysis and the test set analysis. Such a metric is exemplified by the notion of *parse accuracy*, which we define as the percentage of the test sentences for which the most probable analysis (parse) is *identical* to the test set analysis. Although we will also be concerned with a qualitative evaluation of parse results, we believe that an accuracy which is based on an *exact match* is most adequate for a quantitative evaluation.

1.5 Overview of this thesis

In this chapter, we motivated a statistical approach to linguistics from both a psychological and an engineering point of view. We have given the prerequisites of a linguistic performance model which uses the frequencies of sub-analyses in a corpus of sentence-analyses to predict the perceived analysis of a new input sentence. We have proposed an objective method for evaluating performance models, which is the blind testing method combined with an exact match accuracy metric.

In the next chapter, we start with a first realization of a performance model that allows for the use of available language corpora. This first instantiation of the Data-Oriented Parsing framework, which we call DOP1, uses subtrees that occur in the corpus as the "sub-analyses", and uses substitution as the "combination-operation" between subtrees. In order to allow for an estimation of the substitution probabilities of subtrees, two statistical assumptions are used: (1) the subtrees are stochastically independent, and (2), the set of subtrees represents the total population of subtrees. The most important feature of DOP1 is the distinction between the probability of a derivation (of a sentence) and the probability of a parse tree. The probability of a derivation is equal to the product of the substitution probabilities of the subtrees involved, while the probability of a tree is the sum of the probabilities of all distinct derivations generating this tree.

Chapter 3 introduces a Formal Language Theory of Stochastic Grammars in which different stochastic language models can be articulated and compared. We describe DOP1 as a projection of a tree-set into a Stochastic Tree-Substitution Grammar (STSG), and formally compare STSG with Stochastic Context-Free Grammar (SCFG). An important result of this comparison is that SCFGs are stochastically weaker than STSGs: the set of stochastic tree languages generated by SCFGs is a proper subset of the set of stochastic tree languages generated by STSGs. We also compare STSG with two other stochastic grammars that have been proposed to overcome the statistical context-insensitiveness of SCFGs: Stochastic History-Based Grammar and Stochastic Tree-Adjoining Grammar. It turns out that these stochastic grammars do not capture the statistical dependences that can be captured by STSG.

In chapter 4, we deal with the problem of computing the most probable parse of a sentence in DOP1. We distinguish between parsing and disambiguation, showing that the problem does not lie in the creation of a parse forest for an input sentence, but in the selection of the most probable parse from this forest. We show that a so-called Viterbi optimization is not applicable to finding the most probable parse. We propose an iterative Monte Carlo search which estimates the most probable parse with an error that can be made arbitrarily small in polynomial time. Finally, we go into some properties of Monte Carlo disambiguation that are of psychological interest.

In chapter 5, we test the merits of DOP1 as a performance model for syntactic disambiguation. Experiments on part-of-speech strings from the Air-Travel Information System (ATIS) corpus report 96% parse accuracy. If the size of the corpus-subtrees is constrained, the parse accuracy decreases to 52% at subtree-depth one. It turns out that predictions based on the most probable parse are much more accurate than predictions based on the parse generated by the most probable derivation. We also test how much the elimation of once-occurring subtrees affects parse accuracy, and how much the size of the training set affects parse accuracy. Finally, it turns out that DOP1 reaches 100% parse accuracy if training set and test set are united, while SCFG achieves only 59% in that case.

Chapter 6 starts with an investigation of what is involved in extending DOP1 in order to parse sentences that contain unknown words. The model DOP2 is presented as a very simple extension of DOP1: unknown words are labeled by all lexical categories, after which DOP1 is used to generate a most probable parse. Experiments with DOP2 on word strings from the ATIS corpus show, however, a disappointing parse accuracy of 63%. A qualitative evaluation of the test sentences with unknown words indicates that DOP2 has a bias towards using smaller subtrees. The evaluation of the test sentences with only *known* words leads to the notion of "unknown-category word": an ambiguous word which occurs in the training set, but with a different category than is needed to correctly parse the test sentence in which this word appears. It turns out that DOP2 is inadequate for solving unknown-category words.

The outcome of chapter 6 triggers the performance model DOP3; the key insight of DOP3 is that the notion of "unknown subtrees" may overcome the problems with both unknown words and unknown-category words in DOP2. In order to deal with unknown subtrees, we restrict ourselves to subtrees whose unknownness depends only on unknown terminals. The main problem turns out to be the estimation of the frequencies of unknown subtrees. As a solution to this problem, we abolish the assumption that all subtrees have been seen, and treat a corpus as a sample of a larger population. We apply the Good-Turing method for estimating the population probabilities of both unknown and known subtrees. This leads to the definition of the performance model DOP3. Experiments show that DOP3 can quite succesfully parse and disambiguate sentences with unknown (-category) words, yielding 83% parse accuracy for subtree-depth ≤ 3, and that DOP3 does not suffer from the bias towards using smaller subtrees as in DOP2. The Good-Turing method is compared with the Add-k method, resulting in DOP4, which turns out to have a worse parse accuracy than DOP3. In order to get the best possible parse results, DOP3 is finally extended with an external dictionary, yielding the hybrid model DOP5, which achieves an accuracy of 92% (for subtree-depth ≤ 3).

The last chapter of this thesis investigates what is involved in extending a syntactically analyzed corpus with semantic interpretations. We show that if we assume "surface compositionality", the syntactic annotation problem generalizes in a straightforward way to the problem of semantic annotation. However, for analyzing naturally occurring text, partial annotations turn out to be more realistic. We propose the performance model DOP6, in which any corpus-subtree can function as a productive unit, even if its semantics is not yet completely specified, provided that its semantics can be calculated in the end by employing the principle of compositionality in one of two ways: (1) the meaning is constructed by simple composition of the constituents or (2) the meaning is arrived at by abstracting out the contribution(s) of the sister node(s) from the semantics of the node directly governing it. The most important statistical feature of DOP6 is the probability of an interpretation *I* of a sentence as the sum of the probabilities of all the parse trees that have a top node semantics which is logically equivalent to *I*. Finally, as a promise for future research, we go into the influence of discourse structure and recency on the analysis of a new input sentence.

Chapter 2

A First Realization of a Performance Model: The Model DOP1

In this chapter, we describe a first realization of a performance model within the Data-Oriented Parsing framework that allows for the use of currently available corpora. As mentioned in chapter 1, these corpora consist of syntactically labeled phrase structure trees. This means that our performance model will at best be able to account for the syntactic dimension of language perception. Nevertheless, we will see that this first realization of Data-Oriented Parsing, which we will call DOP1, is far from trivial.

2.1 Specifications of DOP1

We make the following choices among the parameters of the DOP framework (sentence-analyses, sub-analyses, combination-operations, combination-probabilities):

(1) sentence-analyses
    syntactically labeled phrase structure trees

(2) sub-analyses
    subtrees

(3) combination-operations
    composition: the identification of the root-node of one subtree with the leftmost non-terminal leaf node of another subtree (in other words: the "substitution" of one subtree on the leftmost non-terminal leaf node of another subtree)

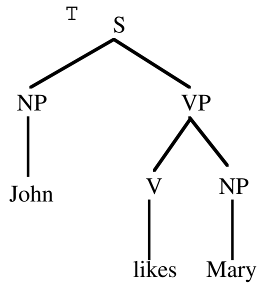
(4) combination-probabilities
    the probability of substituting a subtree on a leftmost non-terminal leaf node of another subtree; this is defined as the probability of selecting a subtree among all corpus-subtrees that could be substituted on a certain non-terminal leaf node

We now give a further explanation of some of these parameters.

Ad (2)

A *subtree* of a tree *T* is a connected subgraph *U* of *T* such that, (1) for every node in *U* holds that it has either zero daughter nodes or all the daughter nodes of the corresponding node in *T*, and, (2) *U* consists of more than one node.

The following example clarifies this definition: in figure 2.1, U_1 and U_2 are subtrees of *T*, whereas U_3 is not. Note that *T* is also a subtree of itself.



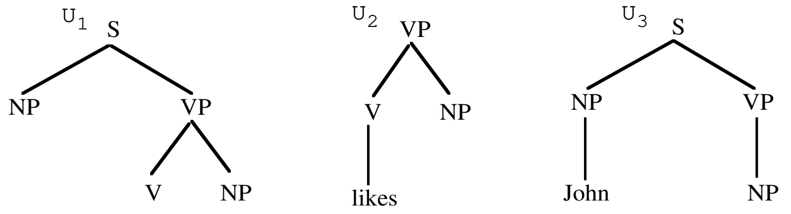


figure 2.1

Ad (3)

The *composition* operation (also called *leftmost substitution* operation) is defined as follows. The composition of subtrees *t* and *u,* $t \circ u$*,* yields a copy of *t* in which its leftmost nonterminal leaf node has been identified with the root node of *u* (i.e., *u* is *substituted* on the leftmost nonterminal leaf node of *t*).

We may define several other combination operations between subtrees, but as a first realization of DOP we will only use this one. Notice the analogy of the substitution operation with rule-application in context-free grammars. The addition of *leftmost* serves to make the composition for two subtrees unique, such that we can speak of a (partial) function. The composition operation is illustrated by figure 2.2:

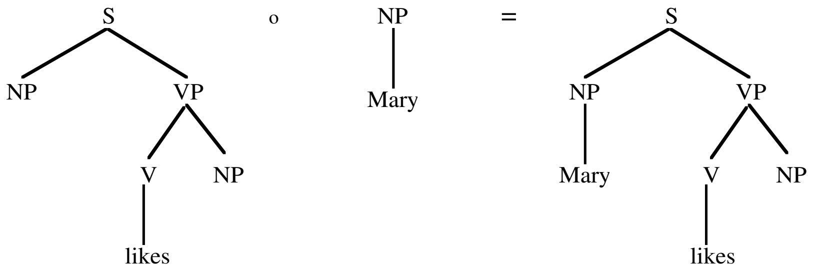


figure 2.2

We will write in the following $(t \circ u) \circ v$ as: $t \circ u \circ v$. Note, however, that composition is not associative. This may be illustrated by figure 2.3.

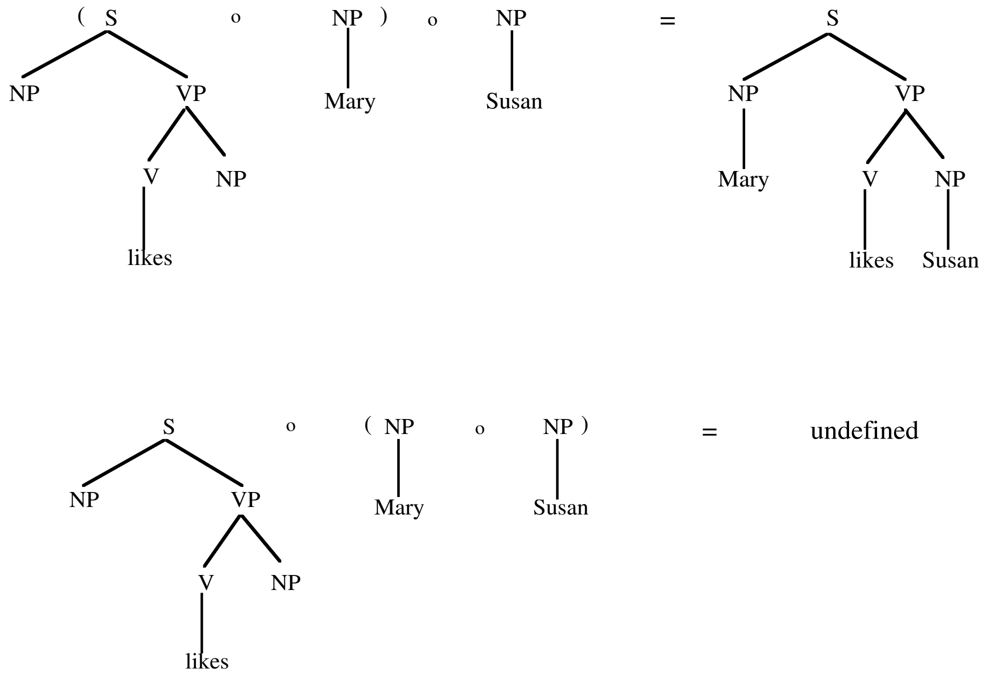


figure 2.3

Ad (4)

For the calculation of the probability of substituting a subtree on a leftmost non-terminal leaf node of another subtree, we use two statistical assumptions: (1) the subtrees are stochastically independent, and (2), the space of corpus-subtrees represents the total population of subtrees. It should be stressed that both assumptions are wrong. However, we will see that assumption (1) is not very harmful if we take into account all, arbitrarily large subtrees; while assumption (2) will be dropped in chapter 7, where a language corpus is treated as a sample of a larger population.

How can we define the probability of combining two subtrees by using the above assumptions? First we observe that due to assumption (1), the probability of substituting a subtree on a leftmost non-terminal node is independent of the subtree in which this leftmost non-terminal node appears. This means that we can simply talk about the substitution probability of a certain subtree given a node of a certain category. Secondly, from our observations in chapter 1, it follows that we need to define the substitution probability of a subtree in terms of its frequency of occurrence in the corpus of sentence-analyses. It is clear that it is not adequate to define a substitution probability as the absolute probability of a subtree in the corpus, but instead as *the probability of a subtree within the space of corpus-subtrees that could be substituted on the same node*. Due to assumption (2), we can compute the substitution probability of a subtree as the ratio between the number of occurrences of this subtree and the total number of occurrences of subtrees that have the same root-category as this subtree. (Note that this implies that the probabilities of the subtrees with the same root-category sum up to one.)

In order to deal with the question as to how the probability of an analysis of an input sentence can be computed from the substitution probabilities of the subtrees that make up the analysis, it is convenient to illustrate DOP1 with an example.

2.2 Illustration of DOP1

We will illustrate DOP1 by means of an extremely simple imaginary example. Suppose that a corpus of sentence-analyses consists of only two trees given in figure 2.4.

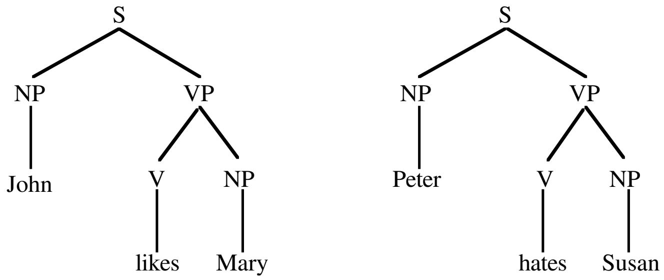
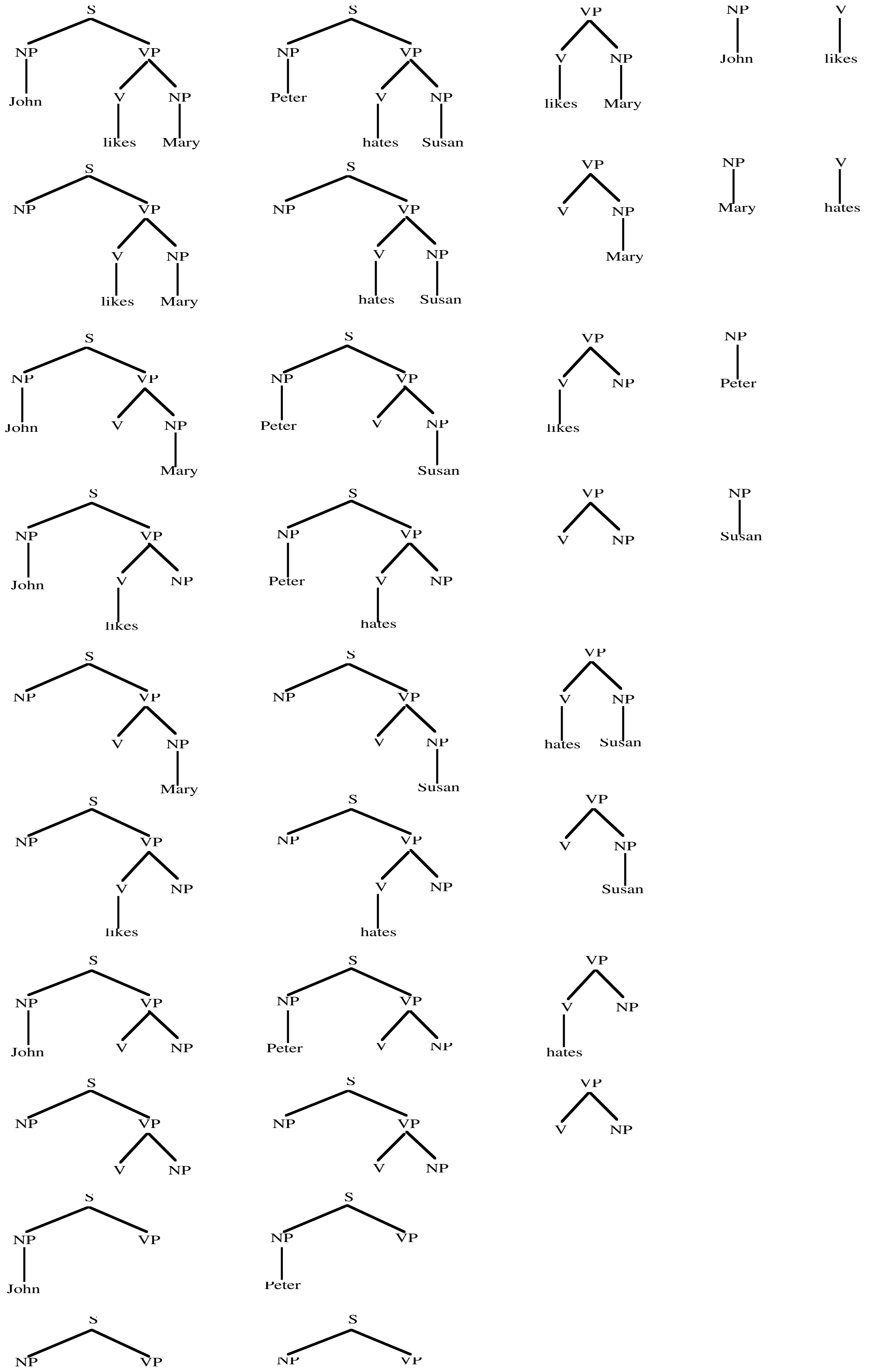


figure 2.4. A corpus of two trees

This means that we have the following multiset of subtrees (figure 2.5):


S NP VP John V NP likes Mary
S NP VP Peter V NP hates Susan
VP V NP likes Mary
NP John
V likes
S NP VP V NP likes Mary
S NP VP V NP hates Susan
VP V NP Mary
NP Mary
V hates
S NP VP John V NP Mary
S NP VP Peter V NP Susan
VP V NP likes
NP Peter
S NP VP John V NP likes
S NP VP Peter V NP hates
VP V NP
NP Susan
S NP VP V NP Mary
S NP VP V NP Susan
VP V NP hates Susan
S NP VP V NP likes
S NP VP V NP hates
VP V NP Susan
S NP VP John V NP
S NP VP Peter V NP
VP V NP hates
S NP VP V NP
S NP VP V NP
VP V NP
S NP VP John
S NP VP Peter
S NP VP
S NP VP

By means of the substitution operation, a new input sentence "*Mary likes Susan*" can be analyzed by combining, for instance, the following subtrees:

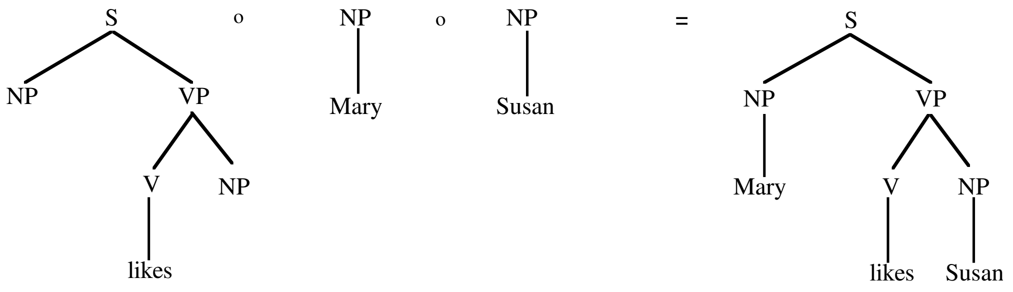


figure 2.6

Such a combination of subtrees will be called a "derivation". We notice that other derivations may yield the same parse tree; for instance:

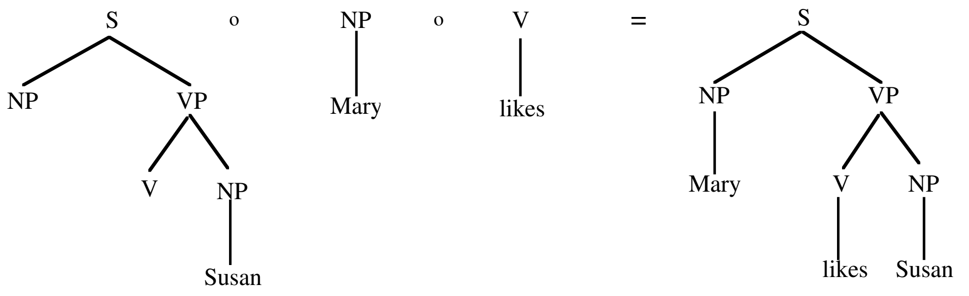


figure 2.7

Or:

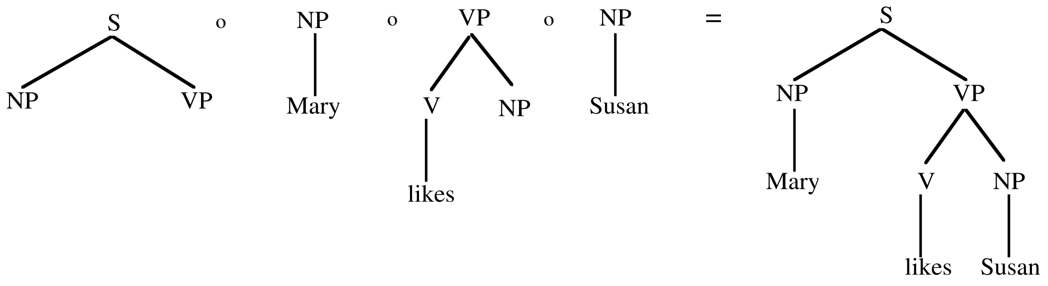


figure 2.8

Thus, a parse tree can be generated by several derivations involving different subtrees.[1] How can we compute the probability of a derivation and that of a parse tree from the substitution probabilities of the subtrees? Let us start with the probability of a derivation, which is the probability that the subtrees are combined by means of substitution. Since we assume that subtrees are stochastically independent, the probability of a derivation is equal to the product of the substitution probabilities of the subtrees.

As an example, we calculate the substitution probabilities of the subtrees in figure 2.8 together with the probability of their resulting derivation. The subtree *S(NP,VP)* occurs twice among a total of 20 subtrees rooted with an *S* (see figure 2.5). Thus, its substitution probability is 2/20.[2] The subtree *NP(Mary)* occurs once among a total of 4 subtrees that can be substituted on the category *NP*, hence, its substitution probability is 1/4. The probability of selecting the subtree *VP(V(likes),NP)* is 1/8, since there are 8 subtrees rooted with a *VP*, among which this subtree occurs once. Finally, the probability of selecting *NP(Susan)* is equal to 1/4. The probability of the resulting derivation is then equal to 2/20 · 1/4 · 1/8 · 1/4 = 1/1280. The next table shows the probabilities of all three derivations given above.

---

[1] We might call this "spurious ambiguity". Note that in this imaginary example, the sentence "*Mary likes Suzan*" is not structurally ambiguous.

[2] The term substitution-probability may seem inappropriate for subtrees rooted with an *S*, since these are not substituted on any other subtree. One should therefore assume a start-symbol *S* on which the first subtree of a derivation is substituted. This will be made rigorous in chapter 3, where we will describe DOP1 as a "stochastic tree-substitution grammar".

P(figure 2.6) = 1/20 · 1/4 · 1/4 = 1/320
P(figure 2.7) = 1/20 · 1/4 · 1/2 = 1/160
P(figure 2.8) = 2/20 · 1/4 · 1/8 · 1/4 = 1/1280

table 2.1

This table shows that a model which defines probabilities over parse trees by taking into account only one derivation, does not accommodate the frequencies of all subtrees that may contribute to the generation of a parse tree. By taking into account the probabilities of all derivations of a parse tree, no subtree that might possibly be of statistical interest is ignored. How can we compute the probability of a parse tree? The probability of a parse is equal to the probability that it is generated by any of its derivations. Since these derivations are mutually exclusive, the probability of a parse is the sum of the probabilities of all its derivations. (This marks the difference with normal "stochastic grammars", where no distinction is made between the probability of a parse tree and the probability of a derivation which generates that tree; cf. §3.3-4). The calculation of the probability of the above parse tree for "*Mary likes Susan*" is left to the reader. Finally, the probability of a sentence or string is the sum of the probabilities of all its parses.

We want to conclude this chapter with an emerging property of DOP1, which will turn out to be of interest for the rest of this thesis. In DOP1, the probability of a parse depends on all derivations that generate that parse; therefore, the more different ways in which a parse can be generated, the higher its probability tends to be; this implies that a parse which can (also) be generated by relatively large subtrees tends to be favored over a parse which can only be generated by relatively small subtrees. Thus, given a sentence, there is a preference for the parse which can be generated by the largest possible subtrees.

Chapter 3

Towards a Formal Language Theory of Stochastic Grammars

In this chapter, we develop a theory in which the properties of stochastic grammars can be formally articulated and compared. We describe DOP1 as a projection of a corpus of tree structures into a Stochastic Tree-Substitution Grammar (STSG), and we formally compare STSG with other stochastic grammars.

3.1 Formal Stochastic Language Theory

The notion of a stochastic grammar usually refers to a finite specification of possibly infinitely many strings and their analyses together with their probabilities. If we want to compare the formal properties of different stochastic grammars, we need a Formal Stochastic Language Theory. In such a theory, we are not so much concerned with weak and strong generative capacity (as is the case in traditional Formal Language Theory), but with weak and strong *stochastic* generative capacity. The following definitions are therefore convenient.

Definitions:

The **stochastic string language** generated by a stochastic grammar *G* is the set of pairs *<x, p(x)>* where *x* is a string from the string language generated by *G* and *p(x)* the probability of that string.

The **stochastic tree language** generated by a stochastic grammar *G* is the set of pairs *<x, p(x)>* where *x* is a tree from the tree language generated by *G* and *p(x)* the probability of that tree.

In analogy to weak and strong equivalence, we define the following equivalences for stochastic grammars:

Definitions:

Two stochastic grammars are called **weakly stochastically equivalent**[1], iff they generate the same stochastic string language.

Two stochastic grammars are called **strongly stochastically equivalent**[2], iff they generate the same stochastic tree language.

Note that if two stochastic grammars are weakly stochastically equivalent they are also weakly equivalent (i.e. they generate the same string language). Moreover, if two stochastic grammars are strongly stochastically equivalent they are also strongly equivalent (i.e. they generate the same tree language) and weakly stochastically equivalent.

---

[1] What we call weak stochastic equivalence is called simply "stochastic equivalence" in (Fu, 1974).

[2] In (Bod, 1993a), this type of equivalence is called "superstrong equivalence".

Now that we have mathematical notions for comparing the generative capacities of stochastic grammars, we want to exclude the pathological cases of *improper* and *infinitely ambiguous* grammars.

Definition Properness of Grammars

A grammar is called *proper* iff only such nonterminals can be generated whose further rewriting can eventually result in a string of terminals.

Example: the context free grammar $<\{S,A\}, \{a\}, S, \{S \rightarrow Sa, S \rightarrow a, S \rightarrow aA\}>$ is not proper, since there is a generation $S \rightarrow aA$ which can never result in a string of terminals.[3]

Definition Finite Ambiguity of Grammars

A grammar is called *finitely ambiguous* if there is no finite string that has infinitely many derivations.

Example: the context free grammar $<\{S\}, \{a\}, S, \{S \rightarrow S, S \rightarrow a\}>$ is not finitely ambiguous, since the string *a* has infinitely many derivations.

Convention
We will only deal with grammars that are proper and finitely ambiguous.

3.2 Stochastic Tree-Substitution Grammar

The way DOP1 combines subtrees into new trees and computes probabilities of derivations and parses may very well be described by what we will call a "Stochastic Tree-Substitution Grammar" (STSG):

---

[3] In (Jelinek et al., 1990), an algorithm is given that determines whether or not a grammar may be made proper by the elimination of rules (p. 31).

Definition Stochastic Tree-Substitution Grammar

A *Stochastic Tree-Substitution Grammar G* is a five-tuple $<V_N, V_T, S, R, P>$ where

V_N is a finite set of nonterminal symbols.

V_T is a finite set of terminal symbols.

$S \in V_N$ is the distinguished symbol.

R is a finite set of elementary trees whose top nodes and interior nodes are labeled by nonterminal symbols and whose yield nodes are labeled by terminal or nonterminal symbols.

P is a function which assigns to every elementary tree $t \in R$ a probability $p(t)$. For a tree t with a root α, $p(t)$ is interpreted as the probability of substituting t on α. We require, therefore, that $0 < p(t) \leq 1$ and $\sum_{t:root(t)=\alpha} p(t) = 1$.

Substitution

If t_1 and t_2 are trees such that the *leftmost nonterminal yield node* of t_1 is equal to the *root* of t_2, then $t_1 \circ t_2$ is the tree that results from substituting t_2 for this leftmost nonterminal yield node in t_1. The partial function $\circ$ is called *leftmost substitution.* We will write $(t_1 \circ t_2) \circ t_3$ as $t_1 \circ t_2 \circ t_3$, and in general $(..((t_1 \circ t_2) \circ t_3) \circ ..) \circ t_n$ as $t_1 \circ t_2 \circ t_3 \circ ... \circ t_n$. For reasons of conciseness we will use the term substitution for leftmost substitution. Notice that the value $p(t)$ for an elementary tree with root α is the probability of substituting t for any nonterminal leaf node α in any elementary tree in *R*.

Derivation

A *leftmost derivation* generated by an STSG *G* is a tuple of trees $<t_1,...,t_n>$ such that $t_1,...,t_n$ are elements of *R*, the root of t_1 is labeled by *S* and the yield of $t_1 \circ ... \circ t_n$

is labeled by terminal symbols. The set of leftmost derivations generated by G is thus given by $Derivations(G) = \{<t_1,...,t_n> \mid t_1,...,t_n \in R \wedge root(t_1) = S \wedge yield(t_1 \circ ... \circ t_n) \in V_T^+\}$. For convenience we will use the term derivation for leftmost derivation. A derivation $<t_1,...,t_n>$ is called a derivation of tree T, iff $t_1 \circ ... \circ t_n = T$. A derivation $<t_1,...,t_n>$ is called a derivation of string s, iff $yield(t_1 \circ ... \circ t_n) = s$. The probability of a derivation $<t_1,...,t_n>$ is defined as $p(t_1) \cdot ... \cdot p(t_n)$.

Parse tree

A *parse tree* generated by an STSG G is a tree T such that there is a derivation $<t_1,...,t_n> \in Derivations(G)$ for which $t_1 \circ ... \circ t_n = T$. The set of parse trees, or *tree language*, generated by G is given by $Parses(G) = \{T \mid \exists <t_1,...,t_n> \in Derivations(G) : t_1 \circ ... \circ t_n = T\}$. For reasons of conciseness we will often use the terms *parse* or *tree* for a parse tree. A parse whose yield is equal to string s, is called a parse of s. The probability of a parse is defined as the sum of the probabilities of all its derivations.

String

A *string* generated by an STSG G is an element of V_T^+ such that there is a parse generated by G whose yield is equal to the string. The set of strings, or *string language*, generated by G is given by $Strings(G) = \{s \mid \exists T : T \in Parses(G) \wedge s = yield(T)\}$. The probability of a string is defined as the sum of the probabilities of all its parses. This means that the probability of a string is also equal to the sum of the probabilities of all its derivations.

It may be evident that STSG is a generalization over DOP1: the model DOP1 projects a corpus of tree structures into an STSG, where the subtrees of DOP1 are the elementary trees of the STSG, and where the substitution probabilities of the subtrees of DOP1 are the probabilities of the corresponding elementary trees of the STSG.

3.3 A Comparison between Stochastic Tree-Substitution Grammar and Stochastic Context-Free Grammar

The oldest and most well-known of all stochastic enrichments of context-free grammars is the so-called "Stochastic Context-Free Grammar" or SCFG (Booth, 1969; Suppes, 1970). An SCFG enriches every rewrite rule of a CFG with a probability which corresponds to the application probability of this rule. In an SCFG, the stochastic dependences are limited to the scope of single rewrite rules. It may be clear that SCFGs run into serious trouble if faced with solving ambiguities that are beyond the scope of single rewrite rules. It is therefore almost evident that SCFGs are stochastically weaker than STSGs. However, as an example of how Formal Stochastic Language Theory may be used to formally articulate this, we will compare SCFG and STSG in the context of this theory. Let us start with the definition of SCFG[4].

Definition Stochastic Context-Free Grammar

A *Stochastic Context-Free Grammar G* is a five-tuple $<V_N, V_T, S, R, P>$ where

V_N is a finite set of nonterminal symbols.

V_T is a finite set of terminal symbols.

$S \in V_N$ is the distinguished symbol.

R is a finite set of productions each of which is of the form $\alpha \rightarrow \beta$, where $\alpha \in V_N$ and $\beta \in (V_N \cup V_T)^+$.

P is a function which assigns to every production $\alpha \rightarrow \beta \in R$ a probability $p(\alpha \rightarrow \beta)$, for which holds that $0 < p(\alpha \rightarrow \beta) \leq 1$ and $\Sigma_x \, p(\alpha \rightarrow x) = 1$.

---

[4] This definition follows (Booth, 1969), (Fu, 1974), (Levelt, 1974), (Wetherell, 1980), (Fujisaki et al., 1989), (Jelinek et al. 1990).

The probability of a leftmost derivation (and its corresponding parse tree) generated by an SCFG is equal to the product of the probabilities associated with the productions applied. Note that, contrary to STSG, every parse tree is generated by exactly one leftmost derivation. The probability of a string generated by an SCFG is equal to the sum of the probabilities of all its derivations.

We will now compare STSG with SCFG in terms of respectively weak and strong stochastic equivalence.

Proposition 1

For every STSG there exists a weakly stochastically equivalent SCFG.

Proof of Proposition 1

Given an STSG G, we convert every elementary tree $t \in R$ into a context-free production $root(t) \rightarrow yield(t)$. This may lead to multiple occurrences of the same production, since different elementary trees may have the same root and yield. To every such production a probability is assigned which is equal to the probability of the tree from which the production is derived. In order to eliminate multiple occurrences of productions, we collapse equivalent productions and add up their probabilities. The resulting SCFG G' generates the same string language as G. It is now easy to see that the sum of the probabilities of all derivations of a string in G is equal to the sum of the probabilities of all derivations of this string in G'. This means that G and G' assign the same probability to every string in their string language. Thus, G and G' are weakly stochastically equivalent.

□

Proposition 2

For every SCFG there exists a weakly stochastically equivalent STSG.

Proof of Proposition 2

Given an SCFG G, we convert every production $\alpha \rightarrow \beta \in R$ into a unique elementary tree t of depth one such that $root(t) = \alpha$ and $yield(t) = \beta$. To every such tree a probability is assigned which is equal to the probability of the production from which the tree is derived. The resulting STSG G' generates the same string language and tree language as G. Now it is easy to see that for every derivation in G there is a unique derivation in G' with the same probability. Thus, the sum of the probabilities of all derivations of a string in G is equal to the sum of the probabilities of all derivations of this string in G'. This means that G and G' assign the same probability to every string in their string language. Thus, G and G' are weakly stochastically equivalent.

□

From the propositions 1 and 2 the following corollary can be deduced.

Corollary 1

The set of stochastic string languages generated by STSGs is equal to the set of stochastic string languages generated by SCFGs.

Corrollary 1 is significant in the sense that if we were only interested in the strings and not in the trees (for instance for the task of string prediction in speech recognition output), we might convert an STSG (and thus a DOP1 model) into a more succinct SCFG.

Proposition 3

For every SCFG there exists a strongly stochastically equivalent STSG.

Proof of Proposition 3

Consider the proof of proposition 2. Since G and G' generate the same tree language and every derivation in G corresponds to a unique derivation in G' with the same probability, G and G' are strongly stochastically equivalent.

□

Proposition 4

There exists an STSG for which there is no strongly equivalent SCFG.

Proof of Proposition 4

Consider the following STSG G consisting of one elementary tree with a probability equal to 1:

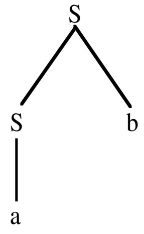


figure 3.1

The tree language generated by G is equal to the set containing only the above elementary tree. An SCFG is strongly equivalent with G if it generates only the above tree. An SCFG which generates the above tree should consist of the productions $S \to Sb$ and $S \to a$. But such an SCFG generates more than just the above tree. Contradiction.

□

Proposition 5
There exists an STSG for which there is no strongly stochastically equivalent SCFG.

Proof of Proposition 5
Consider the proof of proposition 4. Since strong stochastic equivalence implies strong equivalence there is no SCFG which is strongly stochastically equivalent with G.

□

From the propositions 3 and 5 the following corollary can be deduced.

Corollary 2
The set of stochastic tree languages generated by SCFGs is a proper subset of the set of stochastic tree languages generated by STSGs.

Though corollary 2 may seem a significant result, it mainly follows from the property that STSGs are not always strongly equivalent with SCFGs. In the context of stochastic language theory, however, we are not so much interested in tree languages as in *stochastic* tree languages. Thus, it is more interesting to compare stochastic tree languages of strongly equivalent grammars.

Proposition 6

There exists an STSG for which there is a strongly equivalent SCFG but no strongly stochastically equivalent SCFG.

Proof of Proposition 6

Consider the following STSG G consisting of three elementary trees that are all assigned with a probability of 1/3.[5]

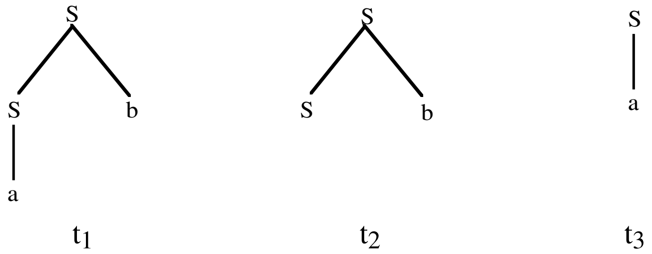


figure 3.2

The string language generated by G is $\{ab^*\}$. The only (proper) SCFG G' which is strongly equivalent with G consists of the following productions.

$S \rightarrow Sb$ (1)
$S \rightarrow a$ (2)

G' is strongly stochastically equivalent with G iff it assigns the same probabilities to the parse trees in the tree language as assigned by G. Let us consider the probabilities of two trees generated by G, i.e. the trees represented by t_1 and t_3.[6] The tree represented by t_3 has exactly one derivation: by selecting the elementary tree t_3. The probability of generating this tree is hence equal to $1/3$. The tree represented by t_1 has two derivations: by selecting elementary tree t_1, or by combining the elementary trees t_2 and t_3. The probability of generating this tree is

---

[5] This STSG is also interesting because it can be projected from a DOP1-model whose corpus of sentence-analyses consists only of tree t_1.

[6] Note that the trees t_1 and t_3 are both elements of the set of (elementary) trees R of G and of the tree language generated by G.

equal to the sum of the probabilities of its two derivations, which is equal to $1/3 + 1/3 \cdot 1/3 = 4/9$.

If G' is strongly stochastically equivalent with G, then it should assign the probabilities 4/9 and 1/3 to the trees represented by t_1 and t_3 respectively. The tree t_3 is exhaustively generated by production *(2)*; thus the probability of this production should be equal to 1/3: $p(S \rightarrow a) = 1/3$. The tree t_1 is exhaustively generated by applying productions *(1)* and *(2)*; thus the product of the probabilities of these productions should be equal to 4/9: $p(S \rightarrow Sb) \cdot p(S \rightarrow a) = 4/9$. By substitution we get $p(S \rightarrow Sb) \cdot 1/3 = 4/9$, which implies that $p(S \rightarrow Sb) = 4/3$. This means that the probability of production *(1)* should be larger than 1, which is not allowed. Thus, G' cannot be made strongly stochastically equivalent with G.

□

The (proof of) proposition 6 is an important result since it shows that STSGs are not only stronger than SCFGs because there are STSGs for which there is no strongly equivalent SCFG, but that STSGs are really *stochastically* stronger, also with respect to SCFGs that might be strongly equivalent to STSGs. It makes also clear why STSGs are stronger: SCFGs cannot attach a probability to a structure larger than one rewrite rule, while STSGs can.

3.4 Other Stochastic Grammars

In this section, we informally compare STSG with two other stochastic language models: Stochastic History-Based Grammar and Stochastic Tree-Adjoining Grammar. These grammars have been proposed as alternatives to SCFG to overcome the stochastic context-insensitiveness of SCFG.

3.4.1 Stochastic History-Based Grammar (SHBG)

Stochastic History-Based Grammars (SHBG[7]) are developed in (Black et al., 1993; Black, Garside & Leech, 1993), though introduced earlier in (Smith, 1973). In SHBG, the probability of applying a rewrite rule in a leftmost derivation is

---

[7] My abbreviation.

made conditional on the rules that were used before in that derivation. In (Black et al., 1993), it is said that SHBG provides "a very rich if not the richest model of context ever attempted in a probabilistic parsing model". However, the limitation to a leftmost derivation for conditionalizing the probability of a rule means that still not all possible stochastic dependences are captured.

Let us illustrate this with the sentence *The emaciated man starved*, of which an analysis is given in figure 3.3. The numbers in the figure refer to the order of applications of the rules in a leftmost derivation of this sentence.

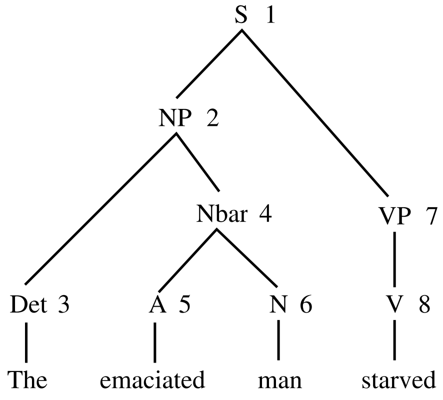


figure 3.3

Suppose that there is a strong stochastic dependence between the words *emaciated* and *starved*, appearing in sentences like the one above, and that these words are largely independent of the words surrounding them (in this case *The* and *man*). An adequate stochastic grammar should be able to account for this specific dependence between *emaciated* and *starved*. It turns out that SHBG is not able to do so. In order to show this, let us explain with somewhat more detail the probabilistic background of SHBG. Suppose that the probability of rule 1 in figure 3.3 (i.e. $S \rightarrow NP\ VP$) is given by $p(1)$. Since in SHGB the rule probability is made conditional on the former rules in the leftmost derivation, the conditional probability of rule 2 is given by $p(2|1)$. The conditional probability of rule 3 is given by $p(3|2,1)$ and so forth. The probability of the whole analysis is equal to the product of the conditional probabilities of the rules: $p(1) \cdot p(2|1) \cdot p(3|2,1) \cdot \ldots \cdot p(8|7,6,5,4,3,2,1)$.

While SHBG can thus capture a dependence between all lexical items *The*, *emaciated*, *man* and *starved* together, there is no way to account for the specific dependence between *emaciated* and *starved*, without *the* and *man*. What would be needed are conditional rule probabilities like $p(8|7,5,4,2,1)$ where the probability of rule 8 is made conditional on all former rules except 6 and 3. SHBG does not account for such probabilities, due to the restriction to a leftmost derivation for conditionalizing the probabilities of rewrite rules. Even if a so-called "finite Markov history" is used, SHBG can only describe the relations between items like *starved* and *man*, *emaciated* and *man*, or *emaciated*, *man* and *starved*, but not between *emaciated* and *starved* alone, since *man* is produced after *emaciated* and before *starved* in a leftmost derivation. Moreover, any restriction to another canonical derivation (rightmost, leftcorner etc.) would yield analogous limitations.

In STSG, on the other hand, the dependence between *emaciated* and *starved* can be captured by an elementary tree in which *emaciated* and *starved* are the only lexical items, and where *the* and *man* are left out, as is shown in figure 3.4.

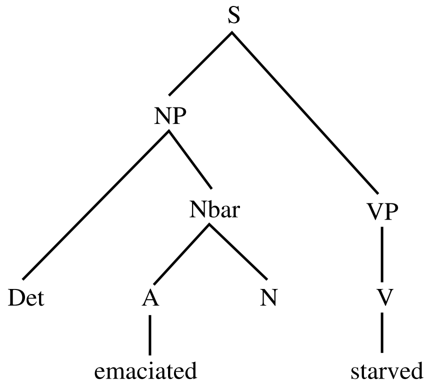


figure 3.4

This artificial example exemplifies a dependence which is strongly semantic in nature. An example which expresses a dependence of a more (semi-)idiomatic nature, is illustrated by the following sentence from the Air Travel Information System (ATIS) corpus (Hemphill et al., 1990): *Show me flights from Dallas to Atlanta.* The NP-construction *flights from X to Y* is almost idiomatic in the ATIS domain and occurs extremely frequently. It may be clear that, analogous to the previous example, SHBG can describe the dependences between all the words of such an NP, but it cannot attach a probability to the NP-construction where *Dallas* and *Atlanta* are left out. This is a serious shortcoming, since for the ambiguity resolution of a sentence which contains an NP like *flights from X to Y*, it is necessary to describe this NP as one statistical unit. STSG, on the other hand, can easily describe this NP as a statistical unit by taking the probability of this construction in the ATIS corpus.

3.4.2 Stochastic Tree-Adjoining Grammar (STAG)

Although STAG (Resnik, 1992; Schabes, 1992) is not a stochastic enrichment of a context-free grammar, but of a tree-adjoining grammar (Joshi, 1987) which belongs to the class of mildy context-sensitive grammars (Joshi et al., 1991), it is interesting to deal with STAG because of its similarity with STSG. An STAG assigns a probability to each elementary (initial or auxiliary) tree that corresponds to the probability that this elementary tree is combined by substitution or adjunction with another elementary tree. If we leave out the adjunction operation, STAG is formally equivalent with STSG. Thus, it looks as if STAG captures at least the stochastic dependences that can be captured by STSG. However, if we look at current instantiations of STAG, we find two serious shortcomings:

(1) Since STAG is linguistically motivated by tree-adjoining grammar (TAG), there are constraints on the form and use of elementary trees. For instance, modifiers are usually represented by separate auxiliary trees, which means that in analyzing the sentence *The emaciated man starved*, the modifier *emaciated* is inserted in the NP *the man* by means of adjunction. Linguistically this may be elegant, but statistically the dependence between *emaciated* and *starved* is lost, since they are not allowed to appear in one elementary tree.

(2) In current implementations of STAG, only the probability of a derivation is accounted for (cf. Resnik, 1992; Schabes, 1992), and not the probability of a resulting tree or of an interpretation. This is statistically inadequate, since, like in STSG, the probability of a derivation is different from the probability of a tree in STAG.

Thus, current instantiations of STAG seem still to be based on the assumption that the statistical dependences coincide with the linguistic dependences of the underlying competence model. In order to create an adequate performance model based on TAG, it is not enough to attach probabilities to the competence units of this model. Instead, competence and performance need to be carefully distinguished, where the performance units must be taken as arbitrarily large trees from a corpus of analyzed language utterances.

3.5 Open problems

There are still many problems to be solved regarding the relations between stochastic grammars. So far, we have only designed the contours of a Formal Stochastic Language Theory which allowed us to formally compare STSG with SCFG. We believe that the following open problems need also to be treated within such a theory (whose solutions fall beyond the scope of this thesis).

* Does there exist a stochastic enrichment of CFG which is stochastically stronger than STSG? We haven't found one yet.

* Is there a stochastic hierarchy within the class of stochastic enrichments of CFGs, where SCFG is at the bottom, STSG at the top, and SHBG somewhere in between?

* If the former question can be answered positively, are there similar stochastic hierarchies in the other classes of the Chomsky hierarchy?

Chapter 4

Computational Aspects of DOP1: Parsing and Disambiguation

In order to deal with the problem of computing the most probable parse tree of a sentence in DOP1 or STSG, we distinguish between parsing and disambiguation. By parsing we mean the creation of a parse forest for an input sentence. By disambiguation we mean the selection of the most probable parse[1] from the forest. The creation of a parse forest is seen as an intermediate step for computing the most probable parse.

4.1 Parsing

From the way STSG combines elementary trees by means of substitution, it follows that an input sentence can be parsed by the same algorithms as (S)CFGs. Every elementary tree *t* is used as a context-free rewrite rule:

---

[1] Note that there can be more than one most probable parse for an input sentence; however, for non-trivial corpora, an input sentence tends to get exactly one most probable parse. In the following, we will assume that the most probable parse is unique.

root(t) → *yield(t)*. Given a cubic time context-free parsing algorithm, an input sentence of length n can be parsed in n^3 time.

Most cubic time parsers make use of a chart or well-formed substring table. A chart parser takes as input a set of context-free rewrite rules and a sentence and produces as output a chart of labeled phrases. A labeled phrase is a sequence of words labeled with a category symbol which denotes the syntactic category of that phrase. A chart-like parse forest can be obtained by including pointers from a category to the other categories which caused it to be placed in the chart. Algorithms that accomplish this can be found in (Younger, 1967; Earley, 1970; Kay, 1980; Winograd, 1983; Jelinek et al., 1990; Schabes, 1991).

In order to obtain a chart-like forest for a sentence parsed by STSG, we need to label the phrases not only with the syntactic categories of that phrase but with the full elementary trees t that correspond to the use of their derived rules *root(t)* → *yield(t)*. Note that in a chart-like forest generated by an STSG, different derivations that generate a same tree do not collapse. We will therefore talk about a *derivation forest* generated by an STSG (cf. Sima'an et al., 1994).

The following formal example illustrates what a derivation forest of a sentence may look like for STSG. In the example, we leave out the probabilities of the elementary trees, that are needed only in the disambiguation process (section 4.2). The visual representation comes from (Kay, 1980): every entry *(i,j)* in the chart is indicated by an edge and spans the words between the i-th and the j-th position of a sentence. Every edge is labeled with the elementary trees that denote the underlying phrase.

The example STSG consists of the following elementary trees:

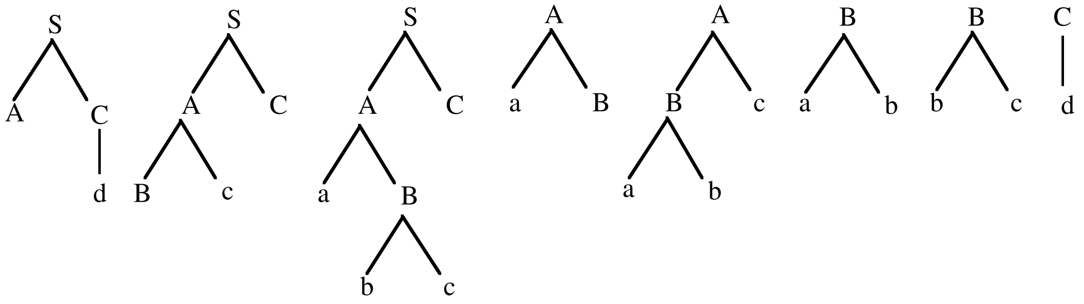


figure 4.1

For the input string **abcd**, the following derivation forest is obtained:

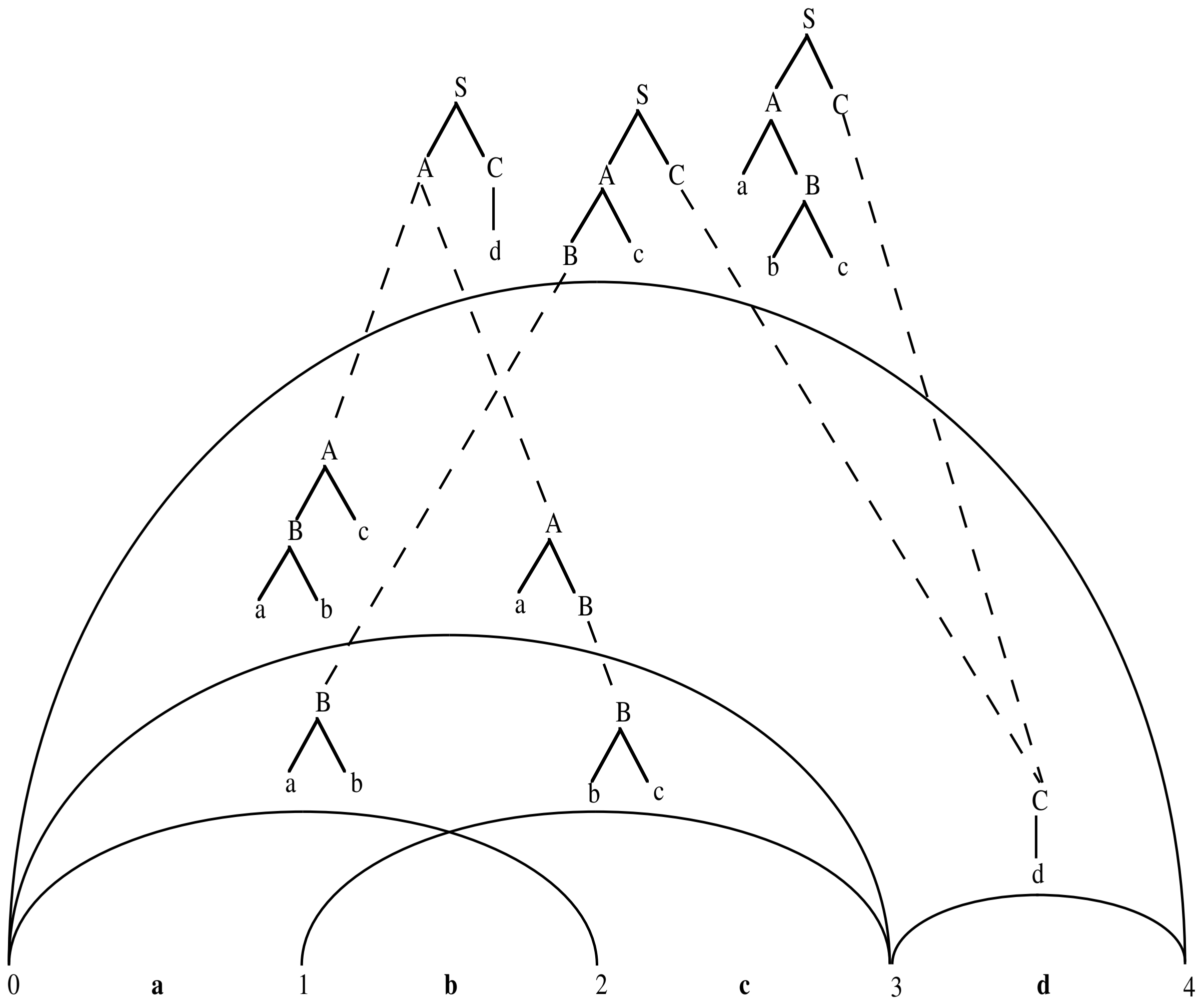


figure 4.2

Note that different derivations in the forest generate the same tree. By exhaustively unpacking the forest, the following trees are obtained:

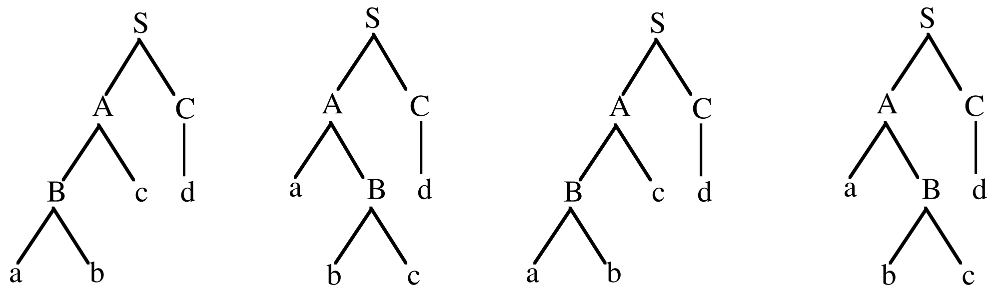


figure 4.3

Thus, every tree is represented twice in the forest by different derivations (with possibly different probabilities). We may ask whether we can pack the forest by collapsing spurious derivations, summing up their probabilities. Unfortunately, no efficient procedure is known that accomplishes this (remember that there can be exponentially many derivations for one tree).

4.2 Disambiguation

Cubic time parsing does not guarantee cubic time disambiguation, as a sentence may have exponentially many parses and any such parse may have exponentially many derivations. Therefore, in order to find the most probable parse of a sentence, it is not efficient to compare the probabilities of the parses by exhaustively unpacking the chart. Even for determining the probability of one parse, it is not efficient to add the probabilities of all derivations of that parse.

4.2.1 Viterbi optimization is not applicable to finding the most probable parse

There exists a heuristic optimization algorithm, known as Viterbi optimization, which selects on the basis of an SCFG the most probable derivation of a sentence in cubic time (Viterbi, 1967; Fujisaki et al., 1989; Jelinek et al., 1990). In STSG, however, the most probable derivation does not necessarily generate the most probable parse, as the probability of a parse is defined as the sum of the probabilities of all its derivations. Thus, there is an important question as to whether we can adapt the Viterbi algorithm for finding the most probable parse.

To understand the difficulty of the problem, we look in more detail at the Viterbi algorithm. The basic idea of the Viterbi algorithm is the pruning of low probability subderivations in a bottom-up fashion. Two different subderivations of the same part of the sentence and whose resulting subparses have the same root can both be developed (if at all) to derivations of the whole sentence in the same ways. Therefore, if one of these two subderivations has a lower probability, then it can be eliminated. This is illustrated by a formal example in figure 4.4. Suppose that during bottom-up parsing of the string *abcd* the following two subderivations *d1* and *d2* have been generated for the substring *abc*. (Actually represented are their resulting subparses.)

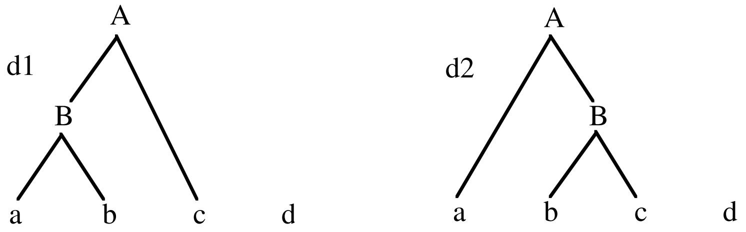


figure 4.4

If the probability of *d1* is higher than the probability of *d2*, we can eliminate *d2* if we are only interested in finding the most probable derivation of *abcd*. But if we are interested in finding the most probable parse of *abcd* (generated by STSG), we are not allowed to eliminate *d2*. This can be seen by the following. Suppose that we have the additional elementary tree given in figure 4.5.

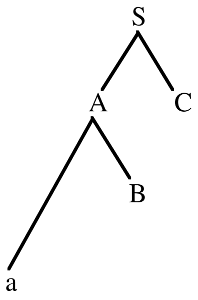


figure 4.5

This elementary tree may be developed to the same tree that can be developed by *d2*, but not to the tree that can be developed by *d1*. And since the probability of a parse tree is equal to the sum of the probabilities of all its derivations, it is still possible that *d2* contributes to the generation of the most probable parse. Therefore we are not allowed to eliminate *d2*.

This counter-example does not prove that there is no heuristic optimization that allows polynomial time selection of the most probable parse. But it makes clear that a "*select-best*" search, as accomplished by Viterbi, is not adequate for finding the most probable parse in STSG. Recent research (Sima'an et al., 1994) has shown that there exists an optimization technique for finding the most probable parse in STSG, but that this technique still takes exponential time. So far, it is unknown whether the problem of finding the most probable parse in a deterministic way is inherently exponential or not. One may of course ask how often the most probable derivation yields in fact the same parse as the computation of the most probable parse. Experiments on the ATIS corpus (see chapter 5) show that this happens only in about 68% of the cases. Moreover, predictions based on the most probable parse are much more accurate than predictions based on the parse generated by the most probable derivation.

4.2.2 Estimating the most probable parse by Monte Carlo search

We will leave it as an open question whether the most probable parse can be deterministically derived in polynomial time. Here we will pursue an alternative approach and ask whether there exists a polynomial time approximation procedure that *estimates* the most probable parse with an error that can be made arbitrarily small.

We have seen that a "*select-best*" search, as accomplished by Viterbi, can be used for finding the most probable derivation in STSG but not for finding the most probable parse. If we apply instead of a *select-best* search, a "*select-random*" search, we can generate a random derivation - provided that the random choices are based on the actual probabilities of the subderivations. By iteratively generating a large number of random derivations we can estimate the most probable parse as the parse which results most often from these random derivations (since the probability of a parse is the probability that any of its derivations occurs). The most probable parse can be estimated as accurately as desired by making the number of random samples as large as desired. According to the Law of Large Numbers, the most often generated parse converges to the most probable parse. Methods that estimate the probability of an event by taking random samples are known as Monte Carlo methods (Meyer, 1956; Hammersley & Handscomb, 1964).[2]

The selection of a random derivation can be accomplished in a bottom-up fashion analogous to Viterbi. Instead of selecting the most probable subderivation at each node-sharing in the chart, a random subderivation is selected at each node-sharing (that is, a subderivation that has *n* times as large a probability as another subderivation should also have *n* times as large a chance to be chosen as this other subderivation). Once arrived at the S-node, the random derivation of the whole sentence can be retrieved by tracing back the choices made at each node-sharing. It trivially follows that the sample probability of a derivation is equal to the probability of that derivation, since the probability of sampling a sequence of subderivations that make up a derivation is equal to the product of the probabilities of sampling the subderivations. We may of course postpone sampling until the S-node, such that we sample directly from the distribution of all S-derivations. But this would take exponential time, since there may be exponentially many derivations for the whole sentence. By sampling bottom-up at every node where ambiguity appears, the maximum number of different subderivations at each node-sharing is bounded to a constant (the total number of

---

[2] Note that Monte Carlo estimation of the most probable parse is more reliable than the estimation of the most probable parse by generating the *n* most probable derivations by Viterbi, since it might be that the most probable parse is exclusively generated by many low probability derivations. The Monte Carlo method is guaranteed to converge to the most probable parse.

rules of that node), and therefore the time complexity of generating a random derivation of an input sentence is equal to the time complexity of finding the most probable derivation, $O(n^3)$. This is exemplified by the following algorithm.

Algorithm 4.1: Sampling a random derivation from a derivation forest

Given a derivation forest, of a sentence of n words, consisting of labeled entries (i,j) that span the words between the i-th and the j-th position of the sentence. Every entry is labeled with linked elementary trees, together with their probabilities, that constitute subderivations of the underlying subsentence. Sampling a derivation from the chart consists of choosing at every labeled entry (bottom-up, breadth-first)[3] at random a subderivation of each root-node:

```
for $k$ := 1 to $n$ do
    for $i$ := 0 to $n-k$ do
        for chart-entry $(i,i+k)$ do
            for each root-node $X$ do
                select at random a subderivation of root-node $X$
                eliminate the other subderivations
```

Let { (e_1, p_1) , (e_2, p_2) , ... , (e_n, p_n) } be a probability distribution of events e_1, e_2, ..., e_n; an event e_i is said to be *randomly selected* iff its chance of being selected is equal to p_i. In order to allow for "simple sampling" (Cochran, 1963), where every event has equal chance of being selected, one must convert a probability distribution into a sample space for which holds that the frequency of occurrence r_i of each event e_i is a positive integer equal to $n \cdot p_i$, where n is the size of the sample space.

We now have an algorithm that selects a random derivation from a derivation forest. Converting this derivation into a parse tree gives a first estimation for the most probable parse. Since one random sample is not a reliable estimate, we need to sample a large number of random derivations and select the parse which results most often from these random derivations. However, we want to have an exact

---

[3] The heuristics of sampling bottom-up, breadth-first can be changed into any other order, such as top-down, depth-first.

criterion for deciding at what number of samples N the most frequently generated parse is a reliable estimate for the most probable parse.

First, we need to have sufficiently small standard errors in the relative frequencies of the parses. We therefore demand that the number of samples N be made sufficiently large in order to make the standard errors sufficiently small (usually 0.05 in sampling experiments). The variance of a parse i with probability p_i is equal to $p_i (1 - p_i)/N$. Since $0 < p_i \leq 1$, the variance is always smaller or equal to $1/(4N)$. Thus, for a standard error σ (which is the square-root of the variance) holds that $\sigma \leq 1/(2\sqrt{N})$. This allows us to calculate a lower bound for N given an upper bound for σ, by $N \geq 1/(4\sigma^2)$. For instance, in order to obtain a standard error of 0.05, N must be at least 100.

A small standard error in the relative frequency of the estimated most probable parse does not guarantee that we have a reliable estimate for the actual most probable parse. For instance, there may be different parses in the top of the sampling distribution that are almost equally likely. We are thus interested in the probability that the most frequently generated parse is not equal to the most probable parse. The upper bound for this probability is given by $\sum_{i \neq 0} (1 - (\sqrt{p_0} - \sqrt{p_i})^2)^N$, where p_0 is the probability of the most probable parse and where p_i are the probabilities of the different parses i; N is the number of Monte Carlo iterations.[4] We will call this upper bound probability the most probable parse error, or simply mpp error. We cannot derive beforehand a lower bound for N as we did for the standard error, but we can estimate the mpp error by replacing p_i by the observed relative frequencies of the parses. Secondly we want to make the mpp error sufficiently small, for instance 0.05, by increasing N. This is not possible if the most probable parse is not unique, and N must be large if there is an i with p_i near to p_0.

Notice that it is not enough to have a small mpp error. We also need sufficiently small standard errors for the estimated parse probabilities. This is exemplified by the following algorithm. (We might also estimate the most probable *derivation* by random sampling; however the most probable derivation can be more effectively generated by Viterbi.)

---

[4] I am most grateful to Chris Klaassen for pointing out this formula to me. The formula is generalizable to the problem of estimating the most frequent type in a population by means of sequential sampling (cf. Hammersley & Handscomb, 1964; Deming, 1966).

Algorithm 4.2: Estimation of the most probable parse (mpp)

Given a derivation forest of a sentence, and thresholds for the standard error and the mpp error:

```
repeat until the standard error and the mpp error are smaller than a threshold
    sample a random derivation from the derivation forest (algorithm 4.1)
    store the parse generated by the sampled derivation
    mpp := parse with maximal frequency
    calculate the standard error of the mpp and the mpp error
```

There is an important question as to how long the estimation of the most probable parse may take. Is there a tractable upper bound on the number of samples N that have to be taken before the standard error and the mpp error are sufficiently small? The answer is yes: from the expressions for the standard error and the mpp error given above, it follows that the worst error in the mpp is quadratic in time cost. Therefore, the worst case time complexity of achieving an error ε is $O(\varepsilon^{-2})$. In practice, this means that in order to reduce ε by a factor of k, the number of samples N needs to be increased k^2-fold.

What is the worst case time complexity of parsing and disambiguation together? That is, given an STSG and an input sentence, what is the maximal time cost of finding the most probable parse of a sentence? If we use a chart parser, the creation of a derivation forest for a sentence of n words takes $O(n^3)$ time. Taking also into account the size G of an STSG (defined as the sum of the lengths of the yields of all its elementary trees), a derivation forest can be created in $O(Gn^3)$ time (cf. Graham, Harrison & Ruzzo, 1980). The time complexity of disambiguation is both proportional to the cost of sampling a derivation, i.e. n^3, and to the cost of achieving a maximal error by means of iteration, which is ε^{-2}. This means that the time complexity of disambiguation is given by $O(n^3\varepsilon^{-2})$. The *total* time complexity of parsing and disambiguation is equal to $O(Gn^3) + O(n^3\varepsilon^{-2}) = O((G+\varepsilon^{-2})n^3)$. Thus, there exists a tractable procedure that estimates the most probable parse of an input sentence. This procedure is linear in the grammar size, quadratic in the error probability, and cubic in the sentence length.

Questions concerning the *actual* time performance of parsing and disambiguation will be dealt with when we discuss the experiments in chapter 5.

Notice that although Monte Carlo disambiguation can estimate the most probable parse of a sentence in polynomial time, it is not in the class of polynomial time decidable algorithms *P*. The Monte Carlo algorithm cannot *decide* in polynomial time what is the most probable parse; it can only make the error probability of the estimated most probable parse arbitrarily small. As such, the Monte Carlo algorithm is a probabilistic algorithm belonging to the class of *B*ounded error *P*robabilistic *P*olynomial time (*BPP*) algorithms (cf. Balcazar et al., 1985). *BPP*-problems are usually characterized as follows: it may take exponential time to solve them exactly, but there exists an iterative estimation algorithm with an error that becomes arbitrarily small in polynomial time.

We hypothesize that Monte Carlo disambiguation is also relevant for other stochastic language models. It turns out that all stochastic extensions of CFGs that are richer than SCFG need exponential time algorithms for finding a most probable parse tree (cf. Carroll & Briscoe, 1992; Black et al., 1993; Magerman & Weir, 1992; Schabes & Waters, 1993). To our knowledge, it has never been studied whether there exist *BPP*-algorithms for these models. Alhough it is beyond the scope of our research, we conjecture that there exists a Monte Carlo disambiguation algorithm for stochastic tree-adjoining grammar.

4.2.3 Psychological relevance of Monte Carlo disambiguation

It is unlikely that people disambiguate sentences by sampling derivations, keeping track of the error probability of the most frequently resulting parse. Nevertheless, we believe there are certain properties of Monte Carlo disambiguation that are of psychological interest. The following lists some of them.

1. Although conceptually Monte Carlo disambiguation uses the total space of possible analyses, it tends to sample only the most likely ones. Very unlikely analyses may only be sampled after considerable time, and it is not guaranteed that *all* analyses are found in finite time. This matches with experiments on human sentence perception where implausible analyses are only perceived with great difficulty after considerable time, and often implausible analyses are not perceived at all (Nicol & Pickering, 1993).

2. Monte Carlo disambiguation does not necessarily give the same results for different sequences of samples, especially if different analyses in the top of the sampling distribution are almost equally likely. In the case there is more than one most probable analysis, Monte Carlo does not converge to one analysis but keeps alternating, however large the number of samples is made. In experiments with human sentence perception, it has often been shown that different analyses can be perceived for one sentence (e.g. MacDonald et al., 1994). And in case these analyses are equally plausible, e.g. due to a lack of disambiguation information, people may perceive so-called fluctuation effects.

3. Monte Carlo disambiguation can be made parallel in a very straightforward way: N samples can be computed by N processing units, where equal outputs are reinforced. The more processing units are employed, the better the estimation. However, since the number of processing units is finite, there is never absolute confidence. In the field of neurobiology and cognitive science, the parallel nature of human information processing is widely recognized.

Chapter 5

Experimental Aspects of DOP1: Disambiguating Part-of-Speech Strings

In this chapter, we test the merits of DOP1 as a performance model for syntactic disambiguation. We report on a number of experiments with an implementation of DOP1 that parses and disambiguates *part-of-speech* strings from the ATIS corpus. As motivated in chapter 1 (section 3), we will limit ourselves to testing whether the most probable parse is a good estimate for the perceived parse according to the corpus, using the blind testing method together with an exact match accuracy metric. In the next chapter, we go into the problem of parsing and disambiguating *word* strings.

5.1 The test environment

To test DOP1, we used a manually corrected version of the Air Travel Information System (ATIS) corpus (Hemphill et al., 1990) annotated in the

Pennsylvania Treebank (Marcus et al., 1993). This corpus is of interest since it is often used by the ARPA community to evaluate their grammars and speech systems (cf. (D)ARPA Proceedings 1989-1994). Moreover, the ATIS corpus is very suitable for our experiments, since it only contains questions and imperatives without discourse structure. The following figure shows a random tree from the corpus annotations (for the annotation guidelines see Santorini, 1990, 1991).

```
((SBARQ
  (WHNP WDT/What
        NNS/flights)
  (SQ  VBP/do
       (NP PP/you)
       (VP VB/have
         (PP (PP IN/from
                 (NP NP/San NP/Francisco))
             (PP TO/to
                 (NP NP/Dallas))))))
?)
```

figure 5.1

The 750 trees from the ATIS corpus were stripped of their words in order to achieve a set of parse trees of part-of-speech sequences. The part-of-speech tags were then marked in order to distinguish them from syntactic categories that bear the same name. For figure 5.1 this yields the following parse tree for the p-o-s string "`WDT NNS VBP PP VB IN NP NP TO NP`".

```
(SBARQ
 (WHNP @WDT
       @NNS)
  (SQ  @VBP
       (NP @PP)
       (VP @VB
         (PP (PP @IN
                 (NP @NP @NP))
             (PP @TO
                 (NP @NP))))))
```

figure 5.2

We used the "blind testing method", as described in chapter 1, dividing the corpus at random into a 90% training set and a 10% test set. The 675 trees from the training set were converted into subtrees together with their substitution probabilities, yielding roughly $4 \cdot 10^5$ different subtrees. The 75 part-of-speech sequences from the test set served as input sentences that were parsed and disambiguated using the subtrees from the training set. As motivated in chapter 1, we use the notion of *parse accuracy* as our accuracy metric, defined as the percentage of the test strings for which the most probable parse is identical to the parse in the test set.

5.2 Accuracy as a function of subtree depth and sample size

It is one of the most essential features of DOP1, that arbitrarily large subtrees are taken into consideration to estimate the probability of a parse. In order to test the usefulness of this feature, we performed different experiments constraining the *depth* of the subtrees. The depth of a tree is defined as the length of its longest path. The following table shows the results of seven experiments for different maximum depths of the training set subtrees[1]. The parse accuracy is rounded off to the nearest integer. The CPU time refers to the average CPU time per string employed by a Spark 2.

depth of subtrees	parse accuracy	CPU time (hours)
1	52 %	.04 h
≤2	87 %	.21 h
≤3	92 %	.72 h
≤4	93 %	1.6 h
≤5	93 %	1.9 h
≤6	95 %	2.2 h
unbounded	96 %	3.5 h

table 5.1

---

[1] Notice that if the depth of subtrees is constrained, their relative frequencies and thus their substitution-probabilities change.

The table shows a dramatic increase in parse accuracy when enlarging the maximum depth of the subtrees from 1 to 2. (Note that for depth one, DOP1 is equivalent to a stochastic context-free grammar.) The accuracy keeps increasing, at a slower rate, when the depth is enlarged further. The highest accuracy is obtained by using *all* subtrees from the training set: 72 out of the 75 sentences from the test set are parsed and disambiguated correctly. Thus, the accuracy increases if larger subtrees are used, though the CPU time increases considerably as well.

It may be relevant to mention that the parse *coverage* was 99%. This means that for 99% of the test strings the perceived parse was in the derivation forest generated by the system. As we have already mentioned in chapter 1, the main task of a performance model is not the generation of all possible parses, but the selection of the perceived parse from the possible parses. Above results show that the most probable parse is a very good prediction for the perceived parse.

It is also interesting to see how the parse accuracy depends on the sample size. In the following figure, parse accuracy is plotted against the first 100 randomly generated derivations, for four different maximum depths: 1, 2, 3 and unconstrained.

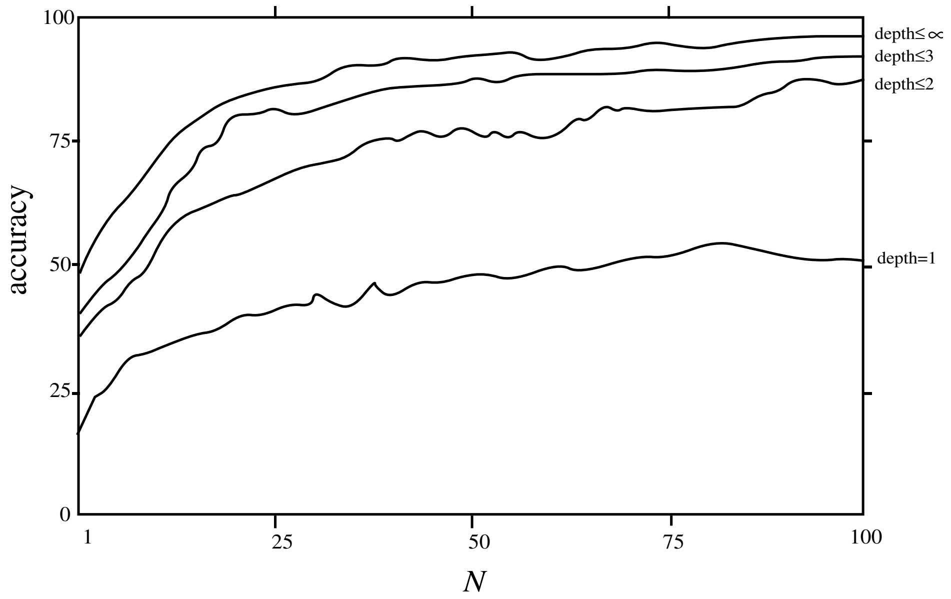


figure 5.3. Parse accuracy against sample size *N* for different subtree-depths

5.3 Does the most probable derivation generate the most probable parse?

Another very important feature of DOP1 is that the probability of a tree is computed as the sum of the probabilities of all different ways of deriving that tree. Although the most probable parse tree of a sentence is not necessarily generated by the most probable derivation of that sentence, there is a question as to how often these two coincide. In order to study this, we also calculated the parse accuracy from the trees generated by the most probable derivations. The following table shows the parse accuracy based on the most probable derivation against the parse accuracy based on the most probable parse, for the 75 test set strings from the ATIS corpus, using different maximum depths for the corpus subtrees.

depth of corpus-subtrees	parse accuracy	
	most probable derivation	most probable parse
1	52%	52%
≤2	47%	87%
≤3	49%	92%
≤4	57%	93%
≤5	60%	93%
≤6	65%	95%
unbounded	65%	96%

Table 5.2. Parse accuracy based on most probable derivation and most probable parse

The table shows that the two parse accuracies are equal if the depth of the subtrees is constrained to 1. This is not surprising, as for depth 1, DOP1 is equivalent with SCFG where every parse is generated by exactly one derivation. What is remarkable, is, that the parse accuracy based on the most probable derivation decreases if the depth of the subtrees is enlarged to 2. If the depth is enlarged further, the accuracy increases again. The highest accuracy by using most probable derivations is obtained by using all subtrees from the corpus (65%), but remains far behind the highest parse accuracy based on the most probable parse (96%). From this table we conclude that predictions based on the most probable parse are much more accurate than predictions based on the parse generated by the most probable derivation. In other words, if we want to predict the appropriate analysis of a string, we should not maximize the probability of the *process* of achieving that analysis but the probability of the *result* of that process.

5.4 The significance of once-occurring subtrees

There is an important question as to whether we need *all* subtrees for accurately predicting the perceived parse of a string, or whether it is possible to eliminate subtrees on statistical grounds, by throwing away very infrequent ones. In order to study this question, we start with a test result. Consider the test sentence "`Arrange the flight code of the flight from Denver to Dallas Worth in descending order`", whose perceived parse according to the test set is:

```
(S (NP *)
   (VP VB/Arrange
       (NP (NP DT/the NN/flight NN/code)
           (PP IN/of
               (NP (NP DT/the NN/flight)
                   (PP (PP IN/from
                           (NP NP/Denver))
                       (PP TO/to
                           (NP NP/Dallas NP/Worth))))))
       (PP IN/in
           (NP (VP VBG/descending)
               NN/order))))
```

figure 5.4

The corresponding p-o-s sequence of this sentence is the test string "`VB DT NN NN IN DT NN IN NP TO NP NP IN VBG NN`". At subtree-depth ≤ 2, the following most probable parse was estimated for this string (where for reasons of readability the words are added to the p-o-s tags):

```
(S (NP *)
   (VP VB/Arrange
       (NP (NP DT/the NN/flight NN/code)
           (PP IN/of
               (NP (NP DT/the NN/flight)
                   (PP (PP IN/from
                           (NP NP/Denver))
                       (PP TO/to
                           (NP NP/Dallas NP/Worth)))
                   (PP IN/in
                       (NP (VP VBG/descending)
                           NN/order)))))))
```

figure 5.5

In this parse tree, we see that the prepositional phrase "`in descending order`" is incorrectly attached to the NP "`the flight`" instead of to the verb "`arrange`". This wrong attachment may be explained by the high relative frequencies of the following subtrees of depth 2 (that appear in structures of sentences like "`Show me the transportation from SFO to downtown San Francisco in August`", where the PP "`in August`" is attached to the NP "`the transportation`", and not to the verb "`show`").

```
NP NP             NP NP
   PP                PP PP
   PP IN                PP
      NP             PP IN
                        NP
```

figure 5.6

Only if the maximum depth was enlarged to 4, subtrees like the following could be used, which led to the estimation of the perceived parse tree.

```
VP VB
   NP NP
      PP
   PP IN
      NP VP VBG
         NN
```

figure 5.7

It is interesting to note that this subtree occurs only once in the training set. Nevertheless, it induces the correct disambiguation of the test string. This seems to contradict the fact that probabilities based on sparse data are not reliable. Since many large subtrees are once-occurring events (*hapaxes*), there seems to be a preference in DOP1 for an occurrence-based approach if enough context is provided: large subtrees, even if they occur once, tend to contribute to the generation of the perceived parse, since they provide much contextual information. Although these subtrees have very low substitution-probabilities, they tend to induce the most probable parse because fewer subtrees are needed to construct a parse.

Additional experiments seemed to confirm this hypothesis. Throwing away all hapaxes yielded an accuracy of 92%, which is a decrease of 4%. Distinguishing between small and large hapaxes, showed that the accuracy was not affected by eliminating the hapaxes of depth 1 (however, as an advantage, the convergence seemed to get slightly faster). Eliminating hapaxes larger than depth 1, decreased the accuracy. The following table shows the parse accuracy after eliminating once-occurring subtrees of different maximum depths.

depth of hapaxes	parse accuracy
1	96%
≤2	95%
≤3	95%
≤4	93%
≤5	92%
≤6	92%
unbounded	92%

Table 5.3. Parse accuracy after eliminating once-occurring subtrees

5.5 Parse accuracy as a function of corpus-size

Given the high parse accuracy achieved by the experiments, we are tempted to conclude that the ATIS corpus is a relatively large corpus for its small domain, where almost all relevant constructions occur. It is interesting to know how the accuracy depends on the size of the corpus. For studying this question, we performed a set of experiments with different training set sizes. Starting with a training set of only 50 trees (randomly chosen from the initial training set of 675 trees), we increased its size with intervals of 50. As our test set, we took the same 75 p-o-s sequences as used in the previous experiments. In the next figure, the parse accuracy is plotted against the training set size.

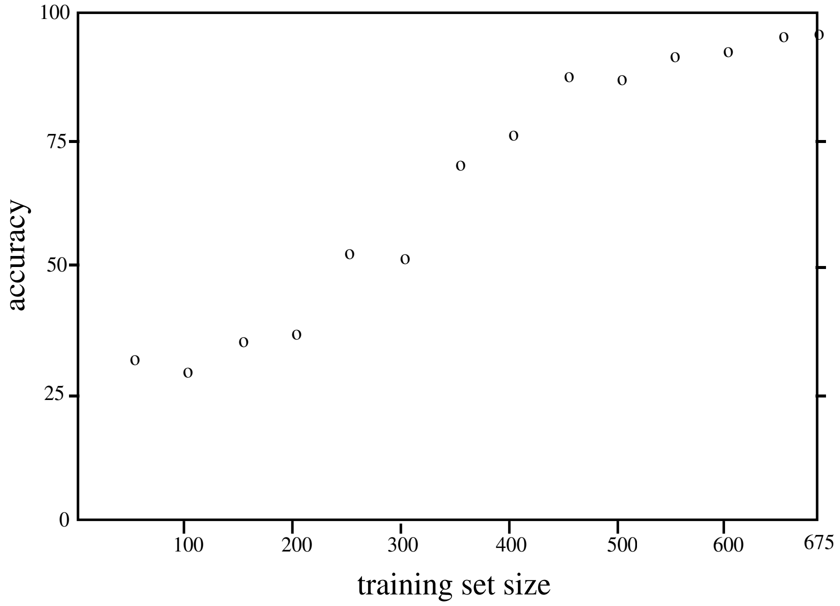


figure 5.8. Parse accuracy against training set size

The figure shows that for a training set of 450 trees, the parse accuracy reaches already 88%. After this, the accuracy grows more slowly, but is still growing at training set size 675. Thus, we might expect an even higher accuracy if we had a larger corpus for this domain.

5.6 Uniting training set and test set. Does it matter?

Our last experiment with p-o-s strings is concerned with the dichotomy of training set and test set. In the tradition of statistical testing, it is considered of utmost importance to keep test set and training set apart. However, as a curious observation, it has been noted that the accuracy of a stochastic parser is not very much affected if the test set is included in the training set. A disappointing observation, since one would expect a sentence to have a very strong preference for getting the same analysis as the analysis of its literal occurrence in the training set. If this does not happen, the underlying stochastic model is almost certainly inadequate for a linguistic performance model.

Although we are well aware of the fact that serious experimenting cannot be accomplished without testing an unseen sample of the domain, we are interested in the performance of DOP1 if test set and training set interfere, because (1), we believe that it is cognitively interesting to see how a sentence which has been experienced before, is analyzed, and (2), because we want to know to what extent the depth of the subtrees affects accuracy if the test set is in the training set.

Therefore we accomplished additional experiments by using the whole ATIS corpus (750 trees) as the training set, while the test set consisted of the same 75 p-o-s strings as in our previous experiments. The following table shows the results of these experiments, where the parse accuracy is given for different maximum depths of the corpus subtrees.

depth of subtrees	parse accuracy
1	59%
≤2	97%
≤3	100%
≤4	100%
≤5	100%
≤6	100%
unbounded	100%

Table 5.4. Parse accuracy against subtree-depth if the test set is in training set

The table shows a relatively low accuracy for depth 1 (59%), which is however larger than when the test set was not in the training set (52%)[2] . If the context is enlarged just a little bit (from depth=1 to depth≤2), there is a very strong preference for the parse tree as it is found in the training set, resulting in an enormous increase in accuracy. If the depth of the subtrees is enlarged any

---

[2] Remember that for subtree-depth 1, DOP1 is equivalent to an SCFG.

further, the parse results match for the full 100%. Thus, the model DOP1 clearly satisfies statement (2) in chapter 1 saying that *people prefer analyses that have been experienced before*.

Chapter 6

A First Extension of DOP1 that Copes with Unknown Words: DOP2

So far, we have shown the adequacy of DOP1 in disambiguating part-of-speech strings. An adequate performance model of natural language should actually be able to predict the perceived analysis of *word* strings. However, word strings derived from a test set often contain words that are unknown in the training set. Since DOP1 only uses subtrees that are literally found in the training set, it cannot adequately parse or disambiguate sentences with unknown words. In this chapter, we start with an investigation of what is involved in extending DOP1 in order to parse sentences that possibly contain unknown words.

There is a basic question as to whether it makes sense to try to parse unknown words with only lexico-syntactic information. It is evident that for *understanding* an unknown word, semantic information is indispensable. However, for the much simpler problem of predicting the syntactic analysis of a sentence with an unknown word, it may be that lexico-syntactic information suffices. Moreover, we want to know how far we can get with only using lexico-syntactic information from a language corpus.

6.1 The model DOP2

As a first possible solution to the problem of unknown words we consider the model DOP2, which is a very simple extension of DOP1: substitute all lexical categories for an unknown word, and estimate the most probable parse of the "sentence" by means of DOP1. The *selected parse* of an input sentence is then defined as the attachment of the unknown words to their corresponding lexical categories in the estimated most probable parse. Thus, unknown words are assigned a lexical category such that their "surrounding" partial parse has maximal probability. We shall refer to this method as the *partial parse method*.

The partial parse method has been used in previous systems that deal with unknown words. A very similar method was applied as early as 1979 in a deterministic parsing system by (Carbonell, 1979), who called it the *project and integrate* method. With the renewed interest in stochastic grammars, several stochastic parsers have adopted the partial parse method in order to obtain some kind of robustness in dealing with unexpected input (e.g. Magerman & Marcus, 1991a, 1991b; Magerman & Weir, 1992; Black et al., 1993).

For its simplicity, the partial parse method may seem attractive. However, it is not founded on a statistical basis, and it therefore remains unclear what its justification is. For instance, the method does not provide probabilities for the *selected parse* of a sentence, but only for the partial parse without the unknown words. Formally, the probability of a selected parse (containing at least one unknown word) is equal to zero. This implies that we cannot estimate the probability of a sentence with this method. It is nevertheless interesting to study where and why the partial parse method can solve unknown words and where and why it cannot. We will see that the outcome of this study leads to a better insight of the properties of unknown words and trigger the development of a better model.

6.2 Formal and computational aspects of DOP2

The backbone of DOP2, which is DOP1, can be formally described by STSG (see chapter 3). However, the attachment of an unknown word to its corresponding lexical category in the most probable partial parse is not described by a stochastic process. Therefore, the generation of a selected parse of a sentence cannot be accomplished by an STSG or any other stochastic grammar. This means that DOP2 cannot be articulated within Formal Stochastic Language Theory, giving it a decisively hybrid character.

As to the computational aspects of DOP2, we can basically employ the same parsing and disambiguation algorithms as developed for DOP1 (see chapter 4). We only need to establish the unknown words of an input sentence beforehand and substitute them by all lexical categories. After estimating the most probable parse, the unknown words are attached to their corresponding lexical categories. Remember that for input sentences without unknown words, DOP2 is formally and computationally equivalent to DOP1.

6.3 Experiments with word strings from ATIS

In our experiments with DOP2, we were interested in testing whether the selected parse is a good estimate for the perceived parse. In doing so, we used the same division of the ATIS corpus as in chapter 5 into a training set of 675 trees and a test set of 75 trees, but now the trees were *not* stripped of their words. As before, we used the notion of *parse accuracy* to indicate the percentage of the test sentences for which the selected parse is identical to the parse in the test set.

The 75 test sentences contained 33 words that were unknown in the training set. This corresponded to 26 test sentences with one or more unknown words and 49 sentences with only known words. In order to study the effect of subtree-depth on the parse accuracy, we performed similar experiments as with part-of-speech strings, constraining the maximum depths of the training set subtrees. For time-cost reasons, no experiments were performed with subtrees larger than depth 3. The following table gives the results of these experiments. We represented for each subtree-depth respectively the parse accuracy of the 26 test sentences that contained at least one unknown word, the parse accuracy of the 49 test sentences without unknown words, and finally, the parse accuracy of all 75 test sentences together.

depth of corpus-subtrees	parse accuracy sentences with unknown words	parse accuracy sentences with only known words	parse accuracy for all sentences
1	15%	39%	31%
≤2	35%	69%	57%
≤3	42%	73%	63%

Table 6.1. Parse accuracy for word strings from the ATIS corpus

We notice that the parse accuracy improves if the maximum depth of the corpus subtrees is enlarged (as is the case with part-of-speech parsing). The increase in accuracy from maximum depth 1 to 2, is even more dramatic than for p-o-s parsing. Focusing on the sentences with unknown words, we see that at subtree-depth ≤ 3 only 42% are parsed correctly. However, if plain DOP1 were used, the accuracy would have been 0% for these sentences.

If we look at the sentences with only known words (where DOP2 is equivalent to DOP1), we see that the maximum parse accuracy of 73% is considerably higher than for sentences with unknown words. However, it remains far behind the 92% parse accuracy of DOP1 for part-of-speech strings (for the same subtree-depth). Word parsing is obviously a much more difficult task than part-of-speech parsing, even if all words are known.

6.4 Evaluation: what goes wrong?

The experimental results suggest the need for a careful evaluation of both the test sentences with unknown words and the test sentences with only known words. We will evaluate the results for subtree-depth ≤ 3, since it is here that highest accuracy is achieved in our experiments.

6.4.1 Test sentences with unknown words

To start with the good news, we show some results of sentences with unknown words that were parsed and disambiguated appropriately by DOP2. The unknown words in the sentences are printed bold.

```
(
 (SQ (VP VBP/Are)
     (NP EX/there)
     (NP DT/any NNS/layovers )
     (PP IN/on
        (NP NN/flight CD/670
            (PP IN/from
               (NP NP/Dallas))
            (PP TO/to
               (NP NP/Boston)))))
?)
```

Figure 6.1. Selected parse for the sentence "Are there any **layovers** on flight 670 from Dallas to Boston?"

```
(
 (SBAR
  (WHNP WDT/What
       (NP NN/keyword ))
  (SQ  VBP/do
      (NP PP/I)
      (VP VB/use
          (NP T)
          (S (NP *)
             TO/to
             (VP VB/initiate
                (NP DT/a NN/purchase
                    (PP IN/of
                        (NP NNS/tickets)))))))
?)
```

Figure 6.2. Selected parse for the sentence "What **keyword** do I use to **initiate** a purchase of tickets?"

```
(
 (S  VB/Please
     (S  (NP *)
       (VP VB/make
           (NP NNS/reservations
             (PP IN/for
                 (NP CD/three))
             (PP IN/on
                 (NP  (NP NP/American NN/flight)
                     (NP NN/number CD/203 )))))))
.)
```

Figure 6.3. Selected parse for the sentence "Please make reservations for three on American flight number **203**."

The results in figures 6.1 through 6.3 seem to indicate that the context of subtrees with depth ≤ 3 can suffice to predict the lexical categories for the unknown words and their appropriate parses.

Let us now consider the bad news, and look at some sentences with unknown words for which the selected parse did not match with the test set parse. The unknown words are given bold again.

```
(   (S
   (NP PP/I)
   (VP VBP/need
       (NP NN/information
           (PP IN/on
               (NP (NP NNS/airlines)
                   (PP IN/servicing
                       (NP NP/Boston))
                   (S (NP *)
                      (VP VBG/flying
                          (PP IN/from
                             (NP NP/Dallas)))))))))
   .)
```

Figure 6.4. Incorrect selected parse for the sentence "`I need information on airlines` **`servicing`** `Boston flying from Dallas.`"

```
( (S
   (NP PP/I)
   (VP VBP/need
       (NP NN/information
           (PP IN/on
               (NP (NP NNS/airlines)
                   (S (NP *)
                      (VP (VP VBG/servicing
                              (NP NP/Boston))
                          (VP VBG/flying
                              (PP IN/from
                                 (NP NP/Dallas))))))))))
   .)
```

Figure 6.5. Test set parse of the sentence "`I need information on airlines` **`servicing`** `Boston flying from Dallas.`"

In the selected parse, resulting from DOP2, the unknown word "`servicing`" is incorrectly tagged as a preposition (`IN`), and consequently attached to a prepositional phrase (`PP`). This may be explained by the very frequent occurrences in the ATIS corpus of prepositional phrases like figure 6.6 and noun phrases like figure 6.7 (both subtrees of depth 3), that led to a most probable partial parse resulting in the selected parse of figure 6.4.

```
PP IN
   NP NP/Boston
```

Figure 6.6

```
NP NP NNS/airlines
   PP IN
      NP NP
```

Figure 6.7

What happened, is, that "`airlines servicing Boston`" was interpreted as something like "`airlines from Boston`". It would be interesting if the incorrect tagging of "`servicing`" could be interpreted as the preposition "`to`", which is distributionally and semantically very similar to "`servicing`". Unfortunately this is not possible, since the Penn Treebank provides the tag "`TO`" for the preposition "`to`", and the tag "`IN`" for all other prepositions. However, the assignment of a PP to the constituent "`servicing Boston`" is semantically correct. The only reason for not annotating "`servicing Boston`" as a PP, is the connection of "`servicing`" with the verb "`service`"; but in order to recognize this in the parsing process we would need morphological annotation or analysis. In the ATIS corpus, "`servicing`" is tagged as a `VBG` (verb gerund) and is attached to a `VP` (figure 6.5).

Another test sentence which was incorrectly parsed is the following:

```
(
 (SBARQ
   (WHNP WDT/Which
         (PP IN/of
             (NP DT/these
                 NN/aircraft NNS/types)))
   (S (NP T)
      (VP VBZ/carries
         (NP DT/the
             NN/fewest
             NN/number)
             (PP IN/of
                (NP NNS/passengers))))))
   ?)
```

Figure 6.8. Incorrect selected parse for "`Which of these aircraft types carries the` **`fewest`** `number of passengers?`"

In this parse, the unknown word "`fewest`" is incorrectly tagged as a noun (`NN`). This may be explained by the very frequent occurrences in ATIS of NPs yielding "`the flight number`", which forces "`fewest`" to be interpreted as a noun. In the parse according to the ATIS corpus, "`fewest`" is tagged as a `JJS` (superlative) and constitutes an adjectival phrase:

```
(
 (SBARQ
   (WHNP WDT/Which
         (PP IN/of
             (NP DT/these
                 NN/aircraft NNS/types)))
   (S (NP T)
      (VP VBZ/carries
         (NP DT/the
             (ADJP JJS/fewest )
             NN/number
             (PP IN/of
                (NP NNS/passengers))))))
    ?)
```

Figure 6.9. Test set parse for "`Which of these aircraft types carries the` **`fewest`** `number of passengers?`"

The following incorrectly parsed test sentence gives a hint of the kind of biases that occur with the partial parse method.

```
(
 (SBARQ (WHNP WDT/What
              (NP NN/movies ))
        (SQ   (NP T)
             VBP/are
             (VP VBN/scheduled  )
             (PP IN/for
                 (NP DT/these NNS/flights))))
   ?)
```

Figure 6.10. Incorrect selected parse for "`What` **`movies`** `are` **`scheduled`** `for these flights?`"

The only error in this parse is the part-of-speech assignment of the unknown word "`movies`". This word is tagged as an NN (noun singular), while it should have been tagged as an NNS (noun plural). The explanation for this may be that in WHNP-subtrees in the ATIS corpus the word "`what`" is more likely to be followed by a singular noun than by a plural noun. This tagging error leads, however, to a disagreement in number between the subject "`movies`", tagged as a singular noun, and the verb "`are`", tagged as a plural verb (VBP). This number-disagreement is striking since there is a large SBARQ-subtree (of depth 3) in the training set that maintains the number-agreement (between a plural subject and the plural verb "`are`" - see figure 6.11). Given the preference in DOP1 for parses that can (also) be contructed by largest possible subtrees (see chapter 2, section 2), one would expect the right tagging of "`movies`". But instead of employing this large SBARQ-subtree, DOP2 employed a smaller WHNP-subtree for parsing and disambiguating the sentence. Evidently, the WHNP-subtree occurs so much more frequently with respect to the SBARQ-subtree, that the most probable partial parse was generated if the WHNP-subtree was used (to which the word "`movies`" was attached for yielding the selected parse).

```
 (SBARQ (WHNP WDT/What
               (NP NNS))
        (SQ   (NP T)
             VBP/are
             VP
             PP))
```

Figure 6.11. SBARQ-subtree from the training set

Something similar seems to have happened in the parsing of the sentence "`What are the amenities?`", where the unknown word "`amenities`" is incorrectly tagged as a singular noun leading once again to a number-disagreement with the verb "`are`" (figure 6.12).

```
( (SBARQ (WHNP WP/What)
         (SQ (NP T)
             (VP VBP/are
                 (NP DT/the NN/amenities))))
  ?)
```

Figure 6.12. Incorrect selected parse for "`What are the` **`amenities`**`?`"

Notice that DOP1 usually tends to satisfy number agreement (even if this agreement is not explicitly coded by means of feature structures in the annotations) because of the preference for parses that can be constructed by larger subtrees. It seems now, that this property cannot be guaranteed by the abberated statistics emerging from DOP2.

6.4.2 Test sentences with only known words: the problem of "unknown-category words"

If we look at the selected parses for the test sentences with only known words, we discover an even more striking result: for several of these sentences no parse could be generated at all, not because a word was unknown, but because a word required a lexical category which it didn't have in the training set. We will call these words "unknown-category words".

To illustrate this, consider the following three test sentences with unknown-category words, of which the first two sentences could not be parsed at all, while the third was parsed inappropriately (the unknown-category words are printed bold):

```
Return to first inquiry.

Where is the stop for USAir flight number 37 from Philadelphia
to San Fransisco?

How much does flight number 83 cost one-way?
```

In the first sentence, the word "`return`" is in the training set only known as a noun (`NN`), whereas its lexical category in this sentence is a verb (`VB`). In the second sentence, the word "`stop`" is only known as a verb (`VB`) in the training set, while its required category is `NN`. In the third sentence, the verb "`cost`" was only known as a noun in the training set, while it should have been a verb.[1] Nevertheless, this last sentence could be parsed, though it led to the following nonsensical most probable parse:

```
(
 (SBARQ
   (WHADVP WRB/How
           RB/much)
   (SQ VBZ/does
       (NP (NP NN/flight NN/number)
           (NP CD/83 NN/cost )
       (ADJP JJ/one-way))
 ?)
```

Figure 6.13. Incorrect selected parse for "`How much does flight number 83 cost one-way?`"

Its parse according to the ATIS corpus is:

---

[1] One might argue that these problems with unknown-category words are due to the tiny size of the ATIS corpus. However, no corpus of any size will ever contain all possible uses of all possible words. Even the extension with a dictionary does not solve the problem. There will be domain-specific words and word senses, abbreviations, proper nouns etc. that are not found in a dictionary. It remains therefore important to know how to deal with unknown words and unknown-category words in a statistically adequate way. The use of the tiny ATIS should just be seen as illustrative for any larger corpus.

```
(
 (SBARQ
  (WHADVP WRB/How
        RB/much)
  (SQ VBZ/does
      (NP (NP NN/flight NN/number)
        CD/83)
      (VP VB/cost )
      (ADJP JJ/one-way)))
?)
```

Figure 6.14. Test set parse for "`How much does flight number 83` **`cost`** `one-way?`"

Notice that richer, morphological annotations would not be of any help here: the words "`return`", "`stop`" and "`cost`" do not have a morphological structure on the basis of which their possible lexical categories can be predicted.

These examples show that we are not only uncertain about the categories of unknown words but also about the categories of known words. We might propose that therefore *all words should be treated as unknown*, such that all words can in principle have any lexical category. However, it is easy to show that DOP2 would give completely erroneous results: if we substitute all words by all lexical categories, every sentence of the same length will get the same most probable parse. And if we treat only the *open-class* words as unknown, we will get this kind of bias for those parts of the sentences that consist of only open-class words.

We may wonder whether it is possible to do better than the partial parse approach. This will be the topic of the next chapter.

Chapter 7

A Corpus as a Sample of a Larger Population: Coping with Unknown Subtrees (DOP3-4-5)

In the previous chapter, we have seen that the partial parse method, employed by DOP2, yields poor predictions for the perceived parse of a sentence with unknown words. For sentences with unknown-category words, the method even appeared to be completely inadequate. DOP2 contained a bias towards using smaller subtrees, which stands in contrast with the preference for parses constructed by largest possible subtrees of DOP1.

A reason for these shortcomings may be the statistical inadequacy of DOP2: it does not allow for the computation of the probability of a parse containing one or more unknown words. As such, the statement in chapter 1, saying that *a comprehender tends to perceive the most probable analysis of a new input*, is not satisfied by DOP2. In this chapter, we study what is involved in creating an extension of DOP1 which allows for the computation of probabilities of parses containing unknown (-category) words. In order to create such an extension, we need a method that estimates the probabilities of unknown *subtrees*.

7.1 The problem of unknown subtrees

By an unknown subtree we mean a subtree which occurs in the test set but not in the training set. In this thesis, we will restrict ourselves to subtrees whose unknownness depends only on unknown terminals. We assume that there are no unknown subtrees that depend on an unknown syntactic structure. Thus, if there is an unknown subtree in the test set, then there is a subtree in the training set which differs from the unknown subtree only in some of its terminals.

Generally posed, this assumption is wrong, although for the ATIS corpus it is not a bad assumption. The high parse accuracy of 96% for part-of-speech strings seems to indicate that almost all syntactic constructions of the ATIS domain are present in the training set. Moreover, the restriction to unknown subtrees containing only unknown terminals may be abolished (although this leads to an impractical space of possible subtrees).

Even with the current restriction, the problem is far from trivial. Questions coming up are:

1. How can we derive unknown subtrees?

2. How can we estimate the probabilities of unknown subtrees?

As to the first question, we are not able to generate the space of unknown subtrees in advance, as we do not know the unknown terminals in advance. But since we assume that all syntactic structures have been seen, we can derive the unknown subtrees that are needed for parsing a certain input sentence, by allowing the unknown words and unknown-category words of the sentence to *mismatch* with the lexical terminals of the training set subtrees. The result of a mismatch between a subtree and one or more unknown (-category) words is a subtree in which the terminals are replaced by the words with which it mismatched. In other words, the subtree-terminals are treated as if they are wildcards. As a result, we may get subtrees in the derivation forest of the sentence that do not occur in the training set.

The mismatch-method has one bottleneck: the unknown words and unknown-category words of a sentence need to be known before parsing can start. It is easy to establish the *unknown* words of a sentence, but it is unclear how to establish the *unknown-category* words. Since every word is a potential unknown-category word (even closed-class words, if a small corpus is used), we ideally need to treat all words of a sentence as possible unknown-category words. Thus, any subtree-terminal is allowed to mismatch with any word of the input sentence. (Later on we will deal with the practical, computational problems that emerge from this approach.)

How can we estimate the probability of a subtree which appears as a result of the mismatch-method in the derivation forest, but not in the training set? It is evident that we cannot assume, as we did in DOP1, that the space of training set subtrees represents the total population of subtrees, since this would lead to a zero probability for any unknown subtree. We therefore drop this assumption, and treat the space of subtrees as a sample of a larger population. We believe that such an approach is also reasonable from a psychological point of view (cf. sections 7.3 and 7.6). An unknown subtree which has a zero probability in the sample may have a non-zero probability in the total population. Moreover, also known subtrees may have sample probabilities that differ from their population probabilities. The problem is how to estimate the population probability of a subtree on the basis of the observed sample. Much work in statistics is concerned with the fitting of particular distributions to sample data, with or without motivating why these distributions might be expected to be suitable. A method which is largely independent of the distributions of population probabilities is the so-called *Good-Turing* method (Good, 1953), which is often used in population biology for estimating the population parameters of species. It only assumes that the sample is obtained at random from the total population. We will propose this method for estimating the population probabilities of unknown subtrees, as well as of known subtrees.

Although Good-Turing is very often used in the field of speech recognition and part-of-speech tagging, it is new in the field of natural language parsing. We will therefore shortly describe the method in the following section (which heavily relies on (Church & Gale, 1991)), after which we will deal with its application to the estimation of the population probabilities of subtrees.

7.2 Good-Turing: estimating the population probabilities of (un)seen types

The Good-Turing method, suggested by A. M. Turing in a 1941 personal communication to I. J. Good (Good, 1953), estimates the expected population probability of a type by adjusting its observed sample frequency. We will refer to r as the observed frequency of a type, and to r^* as the adjusted frequency of r. In order to estimate r^*, Good-Turing uses an additional notion, represented by N_r, which is defined as the number of types which occur r times in an observed sample. Thus, N_r can be seen as the frequency of frequency r. The entire distribution $\{N_1, N_2, N_3, ...\}$ is available in addition to the different rs. The Good-Turing estimator uses this extra information for computing the adjusted frequency r^* as

$$r^* = (r+1) \frac{N_{r+1}}{N_r}$$

Notice that the adjusted frequencies satisfy

$$\sum N_r r^* / N = 1$$

The expected population probability of a type is estimated by r^*/N, where N is the number of observed types. For a more elaborate treatment of Good-Turing, the reader is referred to (Good, 1953). A very instructive paper on the method is found in (Nadas, 1985), who presents three different statistical ways to obtain Turing's formula.

Good-Turing obtains good estimates for r^*/N if N_r is large. We will see that for our applications, N_r tends to be large for small frequencies r, while on the other hand, if N_r is small, r is usually large and needs not to be adjusted.[1]

For the adjustment of the frequencies of unseen types, where $r = 0$, r^* is equal to N_1/N_0, where N_0 is the number of types that we have not seen. N_0 is equal to the difference between the total number of types and the number of observed types.

---

[1] This phenomenon is formalized by Zipf's Law, which states that frequency is roughly proportional to inverse rank (Zipf, 1935).

Thus, in order to calculate the adjusted frequency of an unseen type, one needs to know the total number of types in the population. We will see that for our applications, we can calculate the total number of types by knowing the vocabulary size; this marks a great difference with applications of Good-Turing in for instance population biology, where inferences about the population size are needed.

Notice that Good-Turing does not differentiate among the types that have not been seen: the adjusted frequencies of all unseen types are identical. In the following, we will first use plain Good-Turing, after which we will consider some other methods.

7.3 Using Good-Turing to adjust the frequencies of subtrees

The use of the Good-Turing method in natural language technology is not new. It is commonly applied in speech recognition and part-of-speech tagging for adjusting the frequencies of (un)seen word sequences (e.g. Jelinek, 1985; Katz, 1987; Church & Gale, 1991). In stochastic parsing, Good-Turing has to our knowledge never been tried out. Designers of stochastic parsers seem to have given up on the problem of creating a statistically adequate theory concerning parsing unknown events. Stochastic parsing systems mostly apply (1) a partial parse method (e.g. Magerman & Marcus, 1991a, 1991b; Magerman & Weir, 1992); or (2) a closed lexicon (e.g. Black, Garside & Leech, 1993; Fujisaki et al., 1989); or (3) a two step approach where first the words are tagged by a stochastic tagger, after which the tags are parsed by a stochastic parser. The latter approach has lately become increasingly popular (e.g. Pereira & Schabes, 1992; Schabes, 1993; Weischedel et al., 1993; Briscoe, 1994; Magerman, 1995). However, the tagger used in this two step approach commonly uses Good-Turing for the frequency adjustment of (un)seen n-grams. Why not then directly applying Good-Turing to the structural units of a stochastic grammar? This lack of interest in using Good-Turing may be due to the fact that most stochastic grammars are still being constructed within the *grammar-building* community. In this community, it is generally assumed that grammars need to be as succinct as possible. The existence of unobserved rules is unacceptable from such a (competence) point of view.[2] But from a performance point of view, it is very well acceptable that not all statistical units (in our case, subtrees) have been seen; therefore we will put forward the Good-Turing estimator as a statistically and cognitively adequate extension of DOP1.

How can Good-Turing be used for adjusting the frequencies of known and unknown subtrees? It may be evident that it is too rough to apply Good-Turing to all subtrees together. We must distinguish between subtrees of different roots, since in DOP, the spaces of subtrees of a certain root constitute different distributions, for each of which the substitution-probabilities sum up to one. Therefore, Good-Turing is applied to each subtree-class separately, that is, to the *S*-subtrees, *NP*-subtrees, *VP*-subtrees, *N*-subtrees, *V*-subtrees, *DT*-subtrees, etc. As in the previous chapter, we will take only the subtrees upto depth three.

In order to clarify this, we show in table 7.1 the adjusted frequencies for the class of the 118348 *NP*-subtrees. The first column of the table shows the observed frequencies of *NP*-subtrees from zero to six. The second column shows N_r, the number of *NP*-subtrees that had those frequencies in the training set (the estimation of N_0 is a special case and will be dealt with shortly). The third column shows the adjusted frequencies as calculated by the Good-Turing formula. For instance, for $r = 2$, $N_r = 9057$ and $N_{r+1} = 4161$, thus $r^* = (r+1)\, N_{r+1} / N_r$ = (2+1) 4161/9057 = 1.37. Note that the adjusted frequencies are of the same order as the raw frequencies.

r	N_r	r^*
0	1100000000	0.000055
1	60416	0.30
2	9057	1.37
3	4161	1.86
4	1944	1.99
5	773	3.74
6	482	4.37

Table 7.1. Adjusted frequencies for *NP*-subtrees

---

[2] Although the opposite opinion may be heard as well (e.g. Sampson, 1987).

The calculations for $r = 0$ rest on an estimation of N_0, the number of *NP*-subtrees that have not been seen. N_0 is equal to the difference between the total number of distinct *NP*-subtrees and the number of distinct *NP*-subtrees seen. Thus, we must estimate the total number of possible *NP*-subtrees. To make such an estimation feasible, we use the following assumptions:

* No subtree is larger than depth 3. This was already assumed.

* The unknownness of unseen subtrees only depends on the terminals. Also this was assumed before. It implies that all unlexicalized *NP*-subtrees (i.e. all *NP*-subtrees without words) are known.

* The *size* of the vocabulary is known. This is common practice in corpus linguistics, where the estimations are usually restricted to the domain under study. For instance, for the estimation of the population of bigrams the number of distinct unigrams is usually assumed to be known (Church & Gale, 1991). In our case, we know that the whole ATIS corpus contains 1508 distinct words.[3] (If we did not restrict ourselves to the ATIS domain, we should have assumed a lexicon size of, say, 100.000 words.)

Our calculation of (1) the total number of distinct *NP*-subtrees and (2) N_0 can now be accomplished as follows:

(1) The total number of *NP*-subtrees (that can be the result of the mismatch-method) is calculated by attaching in all possible ways 1508 dummies to the tags of the unlexicalized *NP*-subtrees from the training set. This yields a number of 1.10259×10^9 distinct subtrees. To this number, the number of distinct unlexicalized *NP*-subtrees must be added (12429), yielding 1.10260×10^9 types for the total number of distinct *NP*-subtrees.

(2) The number of unseen types N_0 is the difference between the total number of distinct *NP*-subtrees and the observed number of distinct *NP*-subtrees, $\sum_{r>0} N_r$, which is $1.10260 \times 10^9 - 77479 = 1.10253 \times 10^9$.

---

[3] Remember that this does not mean that we know the word types in the corpus, we only know their *number*.

Notice that N_0 is approximately equivalent to the total number of *NP*-subtrees, which means that hardly any of the possible *NP*-subtrees have been seen. Coming back to our adjustment of the frequency of unseen *NP*-subtrees, this can now be calculated by Good-Turing as $N_1/N_0 = 60416/1.1\text{x}10^9 = 0.000055$. Thus, the Good-Turing method assigns a probability to unseen *NP*-subtrees, as if we had seen each of them 0.000055 times instead of zero times. In order to compensate for moving $1.1\text{x}10^9$ *NP*-subtrees from 0 to 0.000055, the larger frequencies must be adjusted downwards (see table 7.1). In section 7.8, we will study another, easier estimation method for the frequencies of unseen subtrees, which can, however, not be motivated from a statistical perspective.

7.4 The model DOP3

We are now able to give the specifications of DOP3 in terms of our performance model as defined in chapter 1. That is, we instantiate the parameters of the DOP-framework (sentence-analyses, sub-analyses, combination-operations, combination-probabilities):

(1) sentence-analyses

syntactically labeled phrase structure trees

(2) sub-analyses

subtrees

(3) combination-operations

leftmost substitution

(4) combination-probabilities

the probability of substituting a certain subtree on a node of another subtree: this is estimated by the adjusted (relative) frequency of a subtree according to Good-Turing

Notice the resemblance of DOP3 with DOP1. What is different in DOP3 is (1) the much larger space of subtrees, which is extended to subtrees in which one or more terminals are treated as wildcards, and (2) the frequencies of the subtrees, that are now adjusted by the Good-Turing estimator. The probability definitions of a derivation, parse and string in DOP3 are the same as in DOP1. That is, the probability of a derivation is equal to the product of the substitution-probabilities of its subtrees, the probability of a parse tree is equal to the sum of the probabilities of its derivations, and the probability of a string is equal to the sum of the probabilities of its parses.

7.5 Formal and computational aspects of DOP3

It might seem that DOP3 is, like DOP1, formally equivalent to STSG: the subtrees of DOP3 are the elementary trees of an STSG and the combination-probabilities of DOP3 are the probabilities of the elementary trees of an STSG. However, in the traditional notion of "grammar", all structural units (in our case subtrees) need to be defined beforehand. In DOP3, it is impossible to know all subtrees beforehand, since they are only derived during the parsing process by the mismatch-method. Thus, Formal (Stochastic) Language Theory may be appropriate for comparing the formal properties of grammars, it is inadequate for articulating "grammars" that need to deal with unexpected input.

Although DOP3 is not formally equivalent to STSG, this is only due to the rigid definition of the mathematical notion of grammar, where the set of terminals and elementary trees are assumed to be known. In computational practice, we can very easily extend the parsing algorithms designed for STSG to DOP3, by allowing the terminals of subtrees to mismatch with the words of the input sentence. After assigning the adjusted probabilities to the subtrees in the resulting derivation forest, the most probable parse can be estimated by the Monte Carlo algorithms 4.1 and 4.2.

We will not deal here with the practical feasibility of treating all words as unknown (-category) words. Theoretically, the worst case time cost of parsing and disambiguating a sentence in DOP3 is the same as in DOP1, that is, cubic in the sentence length, quadratic in the error and linear in the corpus size. In section

7.7, we will restrict for practical reasons the mismatches to unknown words and "potential" unknown-category words.

7.6 Cognitive aspects of DOP3

Before we go into the experimental behavior of DOP3, we derive some properties of DOP3 that may be of cognitive/linguistic interest.

Preference for parses containing a minimal number of mismatches

The fact that DOP3 assigns very low frequencies to unknown subtrees (that consequently get even lower substitution-probabilities), implies that a most probable parse tends to contain a minimal number of unknown subtrees. Thus, if a parse can be generated by few (or even zero) unknown subtrees, it tends to get a higher probability than a parse which is generated by using many unknown subtrees. This implies that there is a preference for parses constructed by a minimal number of mismatches between subtree terminals and input words. We paraphrase this as: *In DOP3 there is a preference for parses constructed by generalizing over a minimal number of words*.

As an example, consider the ATIS test sentence "*Return to first inquiry*". In chapter 6 (section 4), we have seen that this sentence could not be parsed by DOP2 due to the unknown-category word *return* which is in the training set only known as a noun. In DOP3, this sentence can be parsed if "*return*" mismatches with a subtree-terminal tagged as a verb. However, this sentence can also be parsed if "*first*" mismatches with a noun (such that "*Return to first*" is analyzed as an NP) and "*inquiry*" mismatches with a verb. The latter case would yield a parse which can be generated by derivations containing more unknown subtrees than in the first case, which consequently tends to lead to a lower probability.

Preference for mismatches with open-class words

Another property of DOP3 is that *parses that generalize over open-class words are preferred to parses that generalize over closed-class words*. This can be seen as follows. Closed classes (such as prepositions) normally contain considerably fewer words than open classes (such as nouns). This means that a subtree rooted with a closed-class category tends to have a higher substitution-probability than a subtree rooted with an open class category, since the substitution-probability of a subtree is calculated as the frequency of that subtree divided by the frequency of all subtrees with the same root. For instance, a *P*-subtree will tend to have a higher substitution-probability than an *N*-subtree. Other things being equal, this means that a parse tends to get a higher probability if its subtrees do *not* mismatch with a closed-class word, but with an open-class word. Although this seems only to count for subtrees rooted with lexical categories, there is an actual tendency of DOP3 punishing a mismatch with a closed-class word, since the probability of a parse depends on *all* its derivations.

We believe that this property represents an interesting result, as closed-class words rarely constitute problematic cases in parsing sentences. Note that in DOP3, it is not impossible to generalize over closed-class words, but it will only occur if no parse can be found by generalizing over other words.

As an example, consider again the ATIS test sentence "*Return to first inquiry*". This sentence can be parsed by mismatching "*return*" with a *V*-subtree (i.e. with a verb). But the sentence can also be parsed if not "*return*" but if "*to*" mismatches with a verb. Both cases imply exactly one mismatch with one *V*-subtree. However, the latter case implies a mismatch with a closed-class word, whose *P*-subtree has a relatively high substitution-probability. Thus, the mismatch with "*return*" tends to be preferred, which yields the generation of the correct parse.

7.7 Experimental aspects of DOP3

Although DOP3 is parsable and disambiguatable in polynomial time, the treatment of all words as unknown-category words will certainly lead to an impractically large number of subtrees in the derivation forest. As we have seen, the set of possible *NP*-subtrees (of maximal depth three) consists of 10^9 types, which is a factor 12000 larger than the set of seen *NP*-subtrees (8×10^4). It is therefore evident that we will get impractical processing times in DOP3.

If we still want to perform experiments with DOP3, we need to limit the mismatches as much as possible. It seems reasonable to allow the mismatches only for unknown words, and words that we expect to be potential unknown-category words. From the ATIS training set we derive that only nouns and verbs are actually lexically ambiguous. In our experiments, we will therefore limit the unknown-category words of an input sentence to nouns and verbs. This means that only the words which are unknown in the training set and the words of the sentence which are tagged as a noun or a verb in the training set are allowed to mismatch with subtree-terminals.

Like in our previous experiments, we used the 90%/10% division of the ATIS corpus into a training set of 675 trees and a test set of 75 trees. In order to carefully study the experimental merits of DOP3, we distinguished two classes of test sentences:

1. test sentences containing both unknown and unknown-category words (26)

2. test sentences containing only unknown-category words (49)

Note that all 75 test sentences contained at least one potential unknown-category word (verb or noun). The following table shows the results for subtree-depth ≤ 3, where parse accuracy refers to the percentage of the test sentences for which the most probable parse is identical to the parse in the test set.

test sentences	parse accuracy
with unknown words and unknown-category words	62% (16 out of 26)
with only unknown-category words	94% (46 out of 49)
all test sentences	83% (62 out of 75)

Table 7.2

The table shows that DOP3 has better performance than DOP2 in all respects (cf. table 6.1, page 76). The parse accuracy for the 26 sentences with unknown and unknown-category words is with 62% dramatically higher than the 42% of DOP2. This corresponds to an increase of 5 correctly parsed sentences. For 3 of these 5 sentences, this is due to the correct parsing of unknown-category words, which DOP2 cannot handle anyway. There remain therefore two test sentences that show the actual merits of DOP3 with respect to unknown words. These are represented below (where the unknown words are given bold):

`What are the` **`amenities`**`?`
`What` **`movies`** `are` **`scheduled`** `for these flights?`

We recall that DOP2 tagged, due to a bias for smaller subtrees, both "`amenities`" and "`movies`" incorrectly as singular nouns (NN), thus loosing the agreement with the verb "`are`". DOP3, on the other hand, correctly tagged "`amenities`" and "`movies`" as plural nouns (NNS), thus maintaining the number agreement with "`are`":

```
( (SBARQ (WHNP WP/What)
         (SQ (NP T)
             (VP VBP/are
                (NP DT/the NNS/amenities ))))
 ?)
```

Figure 7.1. Most probable parse for "`What are the` **`amenities`**`?`"

```
((SBARQ (WHNP WDT/What
              (NP NNS/movies ))
        (SQ   (NP T)
             VBP/are
              (VP VBN/scheduled )
              (PP IN/for
                  (NP DT/these NNS/flights))))
   ?)
```

Figure 7.2. Most probable parse for "`What` **`movies`** `are` **`scheduled`** `for these flights?`"

It is fascinating to try to understand why DOP3 assigns the correct tag to "`amenities`", and "`movies`" and the appropriate parses to the whole sentences. Although it is practically impossible to evaluate all the derivations leading to these parses, we observe that the preference for parses that can (also) be contructed by largest possible subtrees (containing agreement), which was lost in DOP2, has been recovered in DOP3.

Nevertheless, there are still 10 sentences with unknown words that are parsed incorrectly by DOP3. It is worth mentioning that for 8 of these sentences DOP3 generated *exactly* the same incorrect parses as DOP2 did, for instance:

```
(  (S
   (NP PP/I)
   (VP VBP/need
       (NP NN/information
           (PP IN/on
               (NP (NP NNS/airlines)
                   (PP IN/servicing
                       (NP NP/Boston))
                   (S (NP *)
                      (VP VBG/flying
                          (PP IN/from
                             (NP NP/Dallas)))))))))
   .)
```

Figure 7.3. Incorrect most probable parse for the sentence "I need information on airlines **servicing** Boston flying from Dallas."

```
(
 (SBARQ
   (WHNP WDT/Which
         (PP IN/of
             (NP DT/these
                 NN/aircraft NNS/types)))
   (S (NP T)
      (VP VBZ/carries
         (NP DT/the
             NN/fewest
             NN/number)
             (PP IN/of
                (NP NNS/passengers))))))
   ?)
```

Figure 7.4. Incorrect most probable parse for "Which of these aircraft types carries the **fewest** number of passengers?"

We conjecture that these sentences can only be parsed correctly if richer morphological annotations are available. In the absence of morphological structure in the ATIS corpus, we shall enrich DOP3 in section 7.9 with an external dictionary.

As to the sentences with only unknown-category words, the improvement of DOP3 with respect to DOP2 is most dramatical: the accuracy increased from 73% to 94%, which is in terms of error reduction, a decrease from 28% to 6%! However, the comparison with DOP2 may not be fair, as DOP2 cannot deal with unknown-category words at all. What the parse results of DOP3 do indicate is, that, for sentences without unknown words, the parse accuracy for word strings is of the same order as the parse accuracy for p-o-s strings (which was 92% at maximum depth 3; see chapter 5).

7.8 Using Add-One instead of Good-Turing: DOP4

There is an intriguing question as to whether the succes of DOP3 depends on the sophisticated Good-Turing estimator, or only on the assignment of very low probabilities to unknown subtrees. Despite the nice statistical justification of Good-Turing, we want to know whether we can achieve the same results if we simply assign a probability to unknown subtrees which is lower than the probabilities of known subtrees. A method which accomplishes this is the so-called *Add-One* method. Although this method has no statistical foundation, it is often used in engineering practice.

According to (Gale & Church, 1994), the first mention of this kind of method is to be found in (Johnson, 1932), who suggests adding some constant k to the frequency r of each type and renormalizing appropriately in order to satisfy $\sum p = 1$. That is, the adjusted frequency, r^*, is $r+k$ times a renormalization factor, $N/(N+kS)$, where S is the number of types. We may call this method the *Add-k* method. The Add-One method is a special case proposed by (Jeffreys, 1948), where $k = 1$. It is defined by $r^* = (r+1)N/(N+S)$. In the field of language technology, this method is common engineering practice; it has been used in adjusting the frequencies of trigrams in a part-of-speech tagger (e.g. Church, 1988), and in adjusting the mutual information for solving *PP*-attachment ambiguities (Hindle

& Rooth, 1993). We will call the modification of DOP3 where the frequencies of the subtrees are adjusted by the Add-One method (or any Add-k method) DOP4.

It is a comfortable property of DOP4 that we do not need to change anything in our parsing and disambiguation algorithms used by DOP3. The only difference is that we adjust the substitution-probabilities of the subtrees in the derivation forest by adding one to the observed frequencies. The most probable parse is estimated by the Monte Carlo algorithms 4.1 and 4.2.

For the experiments, we used exactly the same test-environment as DOP3, assuming that only nouns and verbs can be unknown-category words. The following table compares the results of DOP3 with DOP4.

test sentences	parse accuracy	
	DOP3 (Good-Turing)	DOP4(Add-One)
with unknown words and unknown-category words	62% (16 out of 26)	54% (14 out of 26)
with only unknown-category words	94% (46 out of 49)	84% (41 out of 49)
all test sentences	83% (62 out of 75)	73% (55 out of 75)

Table 7.3

Before going into the test results, we should mention that we are not completely confident about above results: the convergence by Monte Carlo was extremely slow: even at 4000 randomly sampled parses, the mpp error was not sufficiently small. This may be explained by the fact that in the Add-One method, the difference in frequency between known and unknown subtrees is not sufficiently large, such that many more samples are necessary in order to let emerge the most probable parse from the other parses. For time cost reasons, we stopped sampling at 4000.

Table 7.3 suggests that DOP4 has lower accuracy than DOP3 (although it outperforms DOP2). It is disappointing that for the sentences with unknown words, the two problematic sentences which were the flagship of DOP3, were parsed incorrectly by DOP4, i.e. "`What are the` **`amenities`**`?`" and "`What` **`movies`** `are` **`scheduled`** `for these flights?`". Evidently, the problem of adjusting the frequencies of unknown subtrees is not as easy as one may think at first hand.

Above results, together with the slow convergence, suggest that it may be necessary to assign frequencies to unknown subtrees much lower than one, as is also accomplished by Good-Turing. This means that we apply the *Add-k* method instead of *Add-One*. We tried out a number of experiments with different values for *k*, where for $k \leq 0.01$ the convergence by Monte Carlo was relatively fast (at 1500 iterations). The results are given in the following table.

test sentences	parse accuracy	
	DOP3 (Good-Turing)	DOP4(Add-0.01)
with unknown words and unknown-category words	62% (16 out of 26)	54% (14 out of 26)
with only unknown-category words	94% (46 out of 49)	94% (46 out of 49)
all test sentences	83% (62 out of 75)	80% (60 out of 75)

Table 7.4

The table indicates that for the test sentences with only unknown-category words the Add-k method works as well as Good-Turing. However, for the test sentences with also unknown words, DOP4 obtains again the wrong most probable parses for the sentences "`What are the` **`amenities`**`?`" and "`What` **`movies`** `are` **`scheduled`** `for these flights?`". In DOP4, there seems to be an unwanted bias towards smaller subtrees like in DOP2, while we want to have a preference for parses constructed by largest possible subtrees, like in DOP3 and DOP1.

A problem with the Add-k method may be that it does not differentiate between unknown subtrees of different roots. In all subtree classes, the frequencies of unknown subtrees are adjusted by the same constant *k* (after which they are renormalized appropriately). Good-Turing, on the other hand, assigns to unknown subtrees of different roots different frequencies. Although we may try to adapt the Add-k method in the direction of the Good-Turing method, we believe that this would be too much an ad hoc approach and leave the Add-k method for what it is. We hope to have shown that, even if DOP3 uses relatively crude estimations, it is not easy to find a simpler method which obtains the same results.

7.9 Enriching DOP3 with a dictionary: DOP5

An engineering method which may further improve the accuracy of DOP3 does probably not lie in a more reliable frequency estimation of unknown subtrees, but in the use of an external dictionary. In the absence of morphological annotations in ATIS, a dictionary can provide the possible lexical categories of both known and unknown words of an input sentence. For instance, the word "`servicing`" in figure 7.3 will not be considered anymore as unknown and will be assigned with a `VBG` (verb gerund) and an `NN` (singular noun), but not with a `PP` (preposition), which means that the incorrect parse in figure 7.3 cannot be generated anymore.

Unfortunately, the ATIS corpus contains several abbreviations and proper nouns that are not found in a dictionary, and which therefore still need to be treated as unknown by means of DOP3. In the following, we will refer to the extension of DOP3 with an external dictionary as DOP5. DOP5 puts all lexical categories (p-o-s tags) of the sentence words, as found in a dictionary, in the chart. Secondly, the sentence is parsed by DOP3, provided that subtree-terminals are only allowed to mismatch with the words that were not found in the dictionary.

In our experiments, we used Longman Dictionary (Longman, 1988) to assign lexical categories to the words of the test sentences. The lexical categories used in Longman are not equal to the lexical categories used in the ATIS corpus, and needed to be converted. From the 26 sentences with unknown words, 22 could be fully provided with lexical categories. The following table shows the results of DOP5 compared with those of DOP3.

test sentences	parse accuracy	
	DOP3	DOP5
with unknown words and unknown-category words	62% (16 out of 26)	88% (23 out of 26)
with only unknown-category words	94% (46 out of 49)	94% (46 out of 49)
all test sentences	83% (62 out of 75)	92% (69 out of 75)

Table 7.5

The table shows that there is a dramatical increase in parse accuracy from 62% to 88% for sentences with unknown words, while the accuracy for sentences with only unknown-category words remains the same (94%). The total parse accuracy of DOP5 reaches 92%.

We must keep in mind that DOP5 is a hybrid model, where frequencies of subtrees are combined with a dictionary look-up. What we actually need is a much larger and richer annotated corpus. However, we hope to have shown that it is possible to extend DOP1 in a statistically and cognitively adequate way by DOP3, which can parse and disambiguate word strings that contain unknown (-category) words, and which exemplifies the preferable properties of a psychologically plausible model of language performance. If DOP3 is extended by a dictionary, as a surrogate for better corpus annotations, we get a parsing system which can very accurately predict the perceived parse of an input sentence.

Chapter 8

Further Extensions of DOP: Semantics, Discourse, Recency

So far, we have only dealt with the syntactic dimension of language performance, resulting into the implemented systems DOP1 through DOP5. However, no syntactically analyzed corpus can ever stand for a person's past language experience. In this chapter, we investigate what is involved in extending our approach to the problem of semantic interpretation. It must be stressed that this chapter is of informal nature: no system has been implemented, but at the same time no semantically analyzed corpora exist that would allow such a system. At the end of this chapter, we will shortly go into the influence of discourse structure and recency on the probability of analyses.

8.1 Incorporating semantic interpretation[1]

In this section, we consider the problem of computing the semantic interpretation of an input sentence that should be considered the most probable one on the basis

---

[1] The text of this section was partly published in (van den Berg, Bod & Scha, 1994). The author thanks his co-authors for their permission of reproducing it here.

of the frequencies of the interpretations of earlier sentences in a corpus. (We leave aside yet the influence of world knowledge and discourse context on the probability of interpretations.) A performance model for semantic interpretation may be developed after the performance models for syntactic analysis, employing a corpus of sentences with annotations that contain semantic as well as syntactic information. The rest of this section explores this idea. It is divided into two parts. In subsection 8.1.1 we assume that every node in the syntactic constituent structure is annotated with a logical formula expressing its meaning, and that this meaning is derived compositionally from the meanings of its subconstituents. In subsection 8.1.2, we investigate the consequences of dropping this assumption.

8.1.1 Assuming surface compositionality

If we are prepared to assume strict surface compositionality in the semantic annotation of the corpus trees, the performance-based approach to the parsing problem generalizes in a straightforward way to the problem of computing semantic interpretations. By surface compositionality we mean that the way in which the semantics of a surface constituent X correlates with the semantics of the sub-constituents of X can be explicitly indicated: the meaning of X can be specified by a logical expression which contains the meaning representations of the immediate sub-constituents of X as sub-expressions. If this situation consistently applies, we can annotate the corpus trees by (1) specifying for every terminal node a logical formula representing its meaning, and (2) specifying for every non-terminal node a formula schema which indicates how its meaning representation may be put together out of the formulas assigned to its daughter nodes. (In the examples below, these schemata use the variable *d1* to indicate the meaning of the leftmost daughter constituent, *d2* to indicate the meaning of the second daughter constituent, etc.)

Consider a corpus consisting of two very simple sentences, which is annotated in this way, in the following figure. (As logical formulas we use expressions of a standard extensional type theory.)

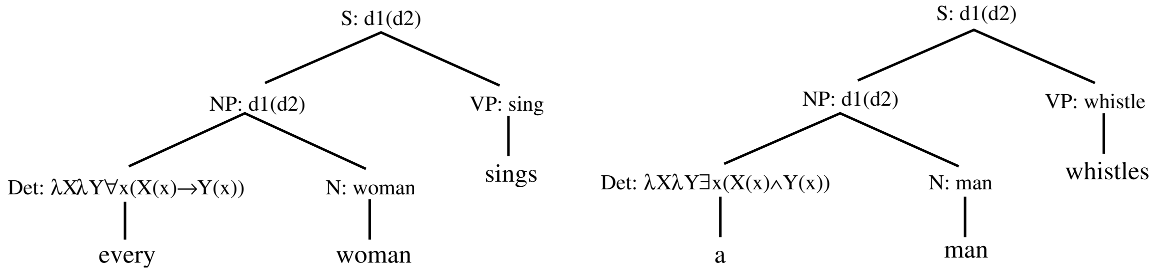
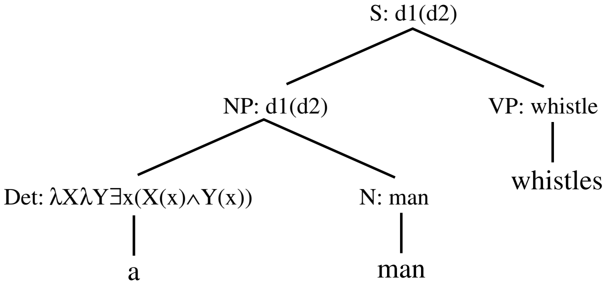


figure 8.1. Imaginary corpus of two trees with syntactic and semantic labels

As in the purely syntactic version of DOP (cf. chapter 2), we now want to compute the probability of an analysis by considering all the different ways in which it can be generated by combining subtrees from the corpus. We can do this in virtually the same way. The only novelty is a slight modification in the process by which a corpus tree is decomposed into subtrees, and a corresponding modification in the composition operation which combines subtrees. If we extract a subtree out of a tree, we replace the semantics of the new leaf node with a unification variable of the same type. Correspondingly, when the composition operation substitutes a subtree at this node, this unification variable is unified with the semantic formula on the substituting tree. (It is required that the semantic type of this formula matches the semantic type of the unification variable.)

A simple example will make this clear. As before, we may try to analyze a sentence like "*a woman whistles*" by combining subtrees from the corpus. Consider the annotated corpus sentence "*a man whistles*" from this point of view. One of the relevant decompositions is the following one:



=

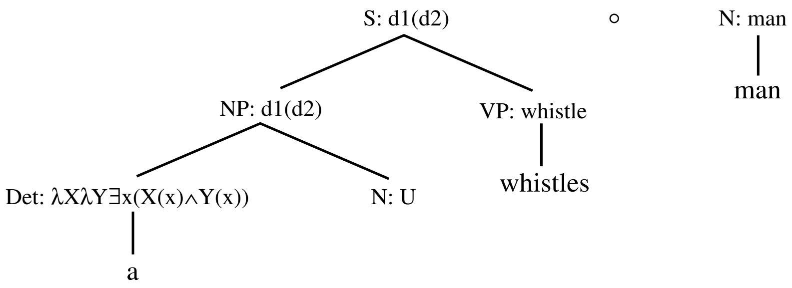


figure 8.2. Decomposing a tree into subtrees with unification variables

We see that by decomposing the tree into two subtrees, the semantics at the breakpoint-node *N: man* is replaced by a variable. Now we can generate an analysis for the sentence "*a woman whistles*" in the following way.

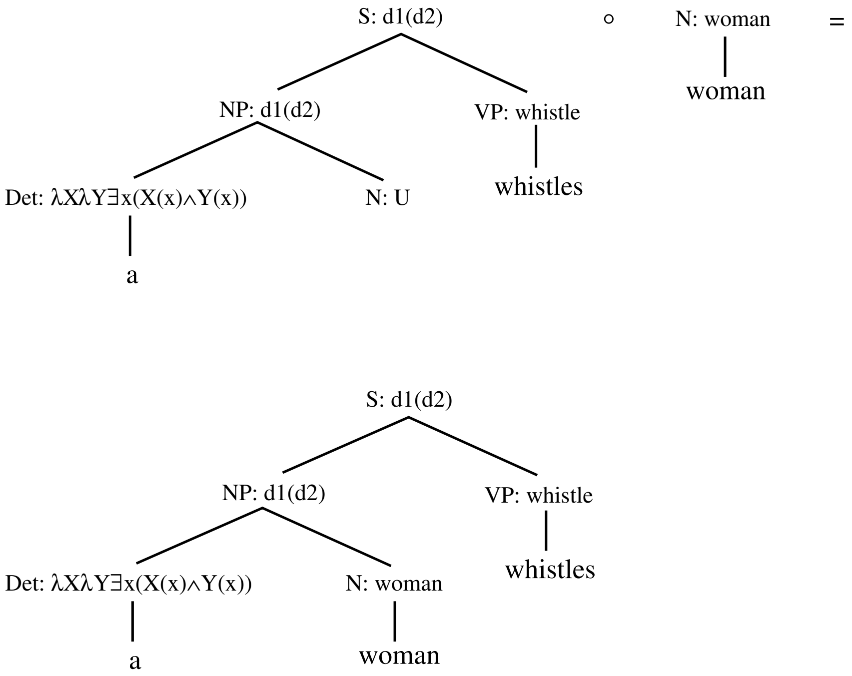


figure 8.3. Generating an analysis for *"A woman whistles"*.

If we would be happy to work with corpora with complete and compositional semantic annotations, we could leave it at that. However, the possibility of annotating large corpora of actually occurring text in this way may be called into question. And it seems that our approach might still work under somewhat less restricted conditions. It is worthwhile, therefore, to explore what happens when we loosen our assumptions about the nature of the semantic annotations.

As a first step, we consider the case where the semantics of the corpus trees and the input sentences *is* in fact compositional, but where we are using an annotation system which does not explicitly presuppose and exploit this. We go through the same simple example, but assuming a different annotation regime. Thus, consider again a corpus consisting of the same sentences as in figure 8.1, but now with the following annotation: every constituent carries a label that consists of a syntactic category and a logical formula representing the meaning of the constituent.

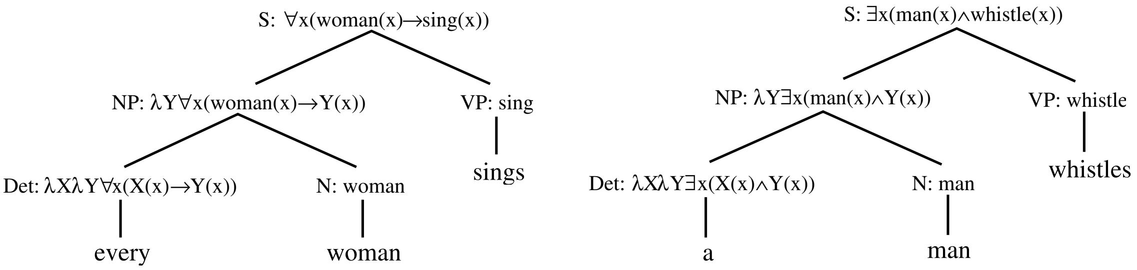


figure 8.4. Imaginary corpus of two trees with syntactic and semantic labels.

Again, we want to compute the probability of an analysis by considering all the different ways in which it can be generated by combining subtrees from the corpus. But decomposing the corpus trees into subtrees is a bit more complex now. This is due to the fact that possible dependencies between the semantic annotations of different nodes are not explicitly indicated now, while they may be strongly suggested by the fact that the formulas on these nodes contain the same descriptive constants.

To deal with this issue, the decomposition process by which subtrees are derived from the initial corpus of trees, may now be further refined as follows. If we extract a subtree out of a tree, we replace the semantics of the new leaf node with a unification variable (of the same type), and introduce this unification variable at every place where the semantics of this subtree appears in the labels of its governing nodes. The same example discussed before may serve to illustrate this.

Again, we try to analyze the sentence "*a woman whistles*" by combining subtrees from the corpus. Consider the annotated corpus sentence "*a man whistles*". One of the relevant decompositions is the following:

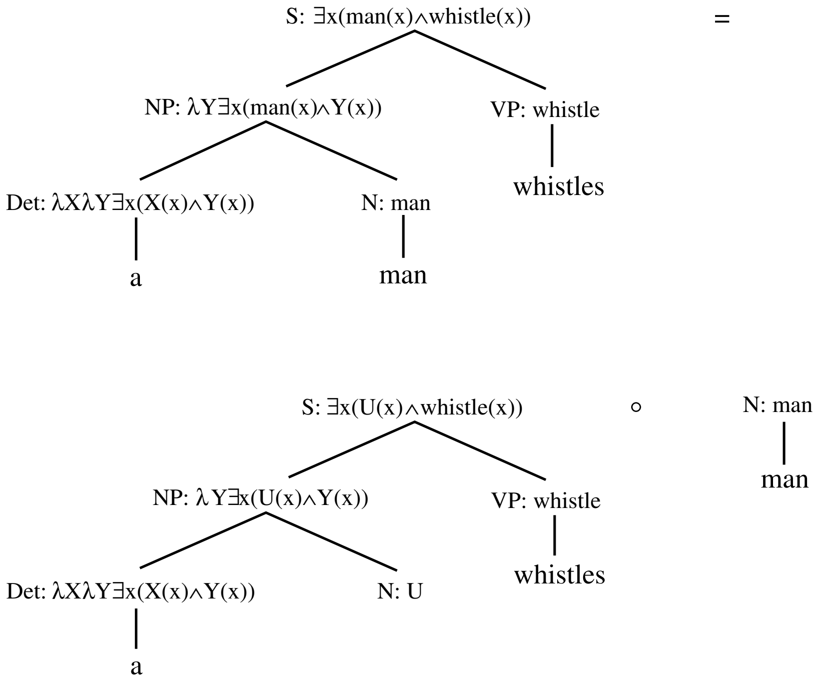


figure 8.5. Decomposing a tree into subtrees with unification variables

We see that by decomposing the tree into two subtrees, the semantics at the breakpoint-node *N: man* is abstracted out of this label and out of the meanings of its governing constituents by replacing it with a variable. If we want to compose these two subtrees again, we need to enrich our composition operation with a unification mechanism: the meaning of the substituted subtree is unified with the variable in the label on which this subtree was substituted, and with the corresponding variables in the governing labels. Having this new combination operation and our abstraction mechanism, we can generate an analysis for the sentence "*a woman whistles*" in the following way.

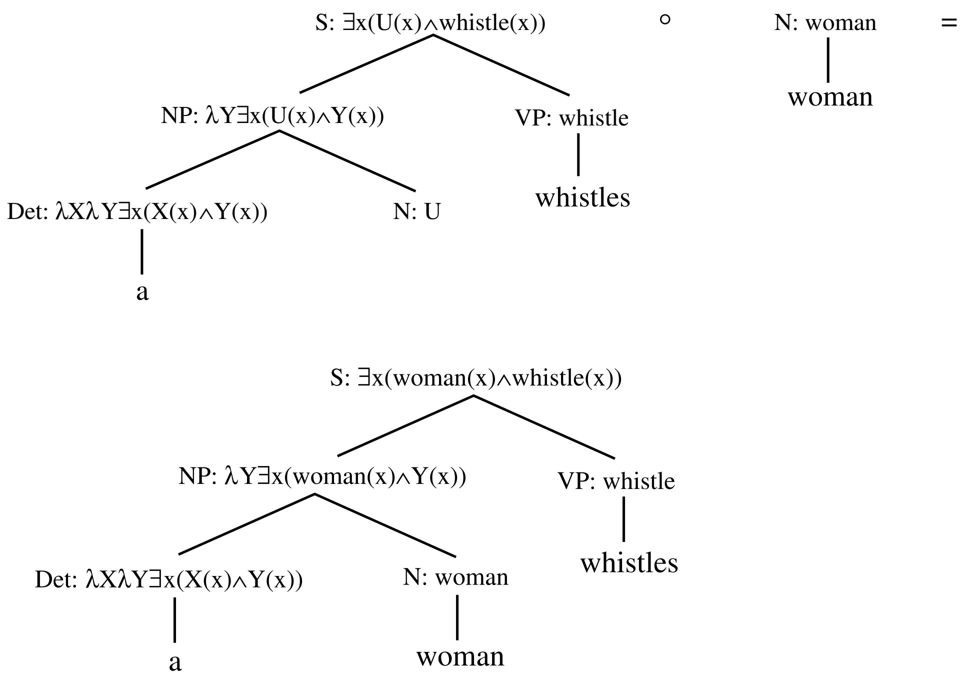


figure 8.6. Generating an analysis for *"a woman whistles"* with abstraction and unification

Of course, this example was unrealistically simple. It may often happen that the logical expression representing the meaning of a constituent is not literally present in the meaning representations of the larger constituents that contain it. In such cases the meaning of a subtree cannot be abstracted directly out of the formulas on its governing nodes. However, if we in fact assume surface compositionality, the class of expressions *equivalent* to the formula on a certain governing node, will always contain expressions in which the semantics of a subordinate constituent *does* literally (and non-trivially) appear. Let us consider the following tree again:

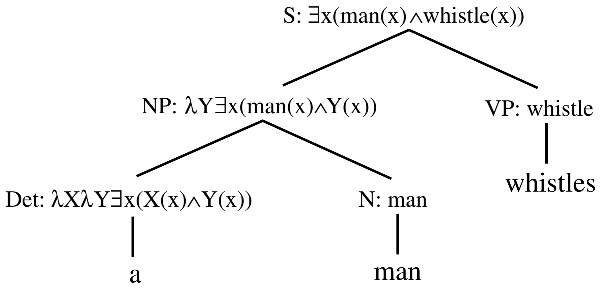


figure 8.7.

We note that the representation of the meaning of the NP "*a man*", which is $\lambda Y\exists x(man(x)\wedge Y(x))$, is not literally present in the logical expression at the S node, which is $\exists x(man(x)\wedge whistle(x))$. However, in the class of expressions equivalent to $\exists x(man(x)\wedge whistle(x),$ there is the expression $\lambda Y\exists x(man(x)\wedge Y(x))\ (whistle)$ out of which the expression $\lambda Y\exists x(man(x)\wedge Y(x))$ can be directly abstracted, yielding $U(whistle)$. Assuming an algorithm for computing equivalent expressions, we may thus be able to decompose the above tree into the following subtrees:

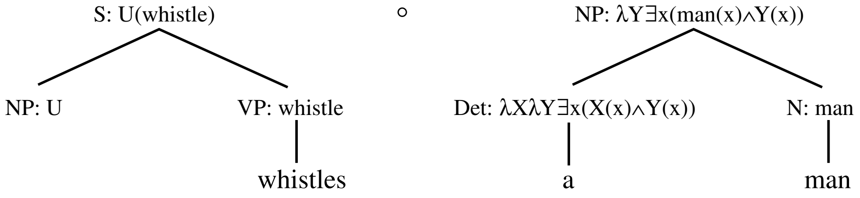


figure 8.8. Decomposing the tree in figure 8.7 into two subtrees by using equivalence relations

And we can now parse the sentence "*every woman whistles*" by combining the following subtrees from the corpus.

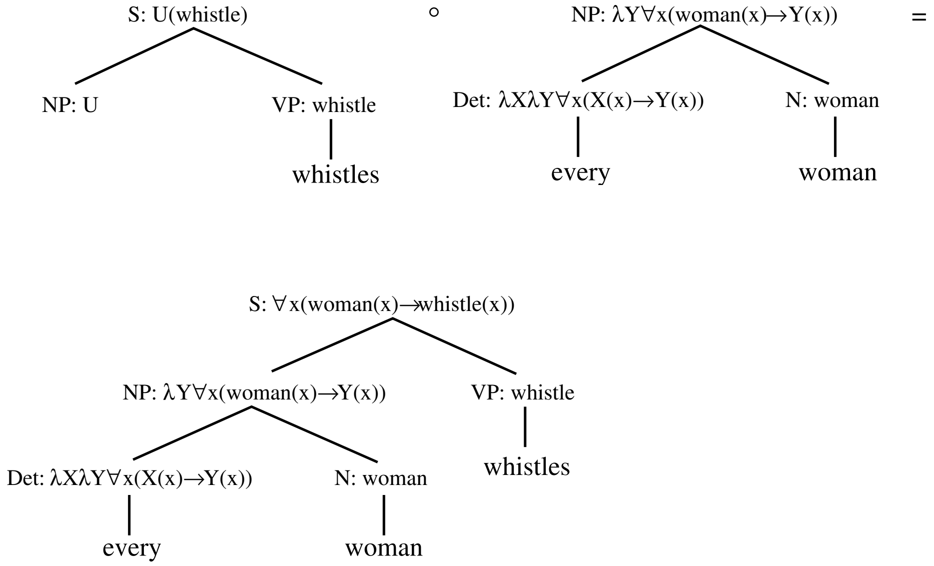


figure 8.9. Parsing the sentence *"every woman whistles"*

It is perhaps trivial to note that, analogous to DOP1, a tree can be decomposed also into more than two subtrees, as is illustrated in figure 8.10 (we leave the generation of the total space of subtrees to the reader).

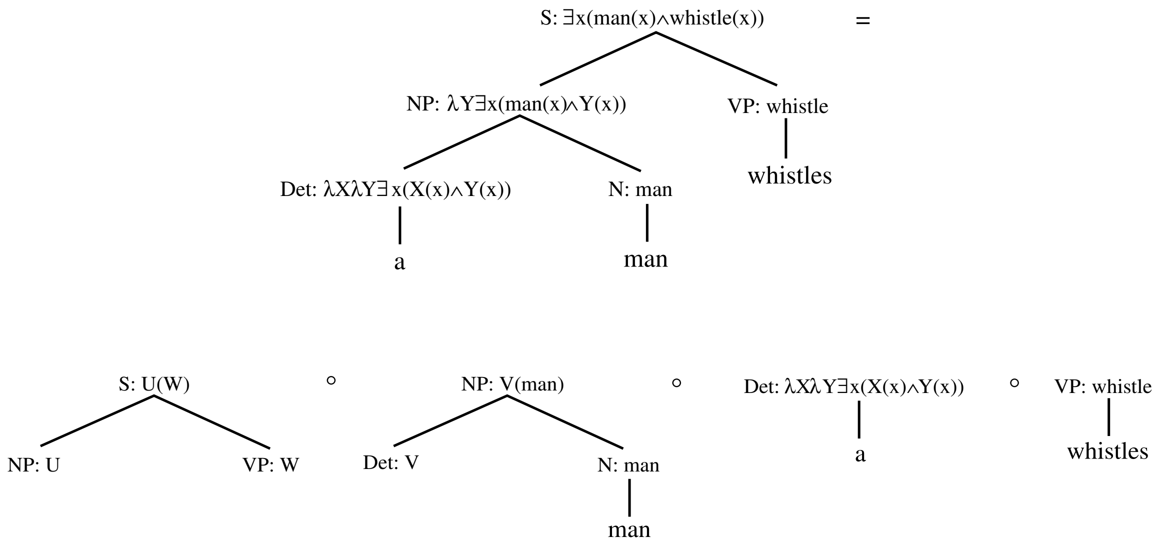


figure 8.10. A tree can be decomposed into several subtrees

We have thus seen that it is not impossible to work with a corpus that is annotated in a way which downplays surface compositionality in that constituents are labelled with fully instantiated semantic representations, rather than with definitions that refer to the semantics of their sub-constituents. We have also seen that there may be a certain cost: this approach may give rise to substantial amounts of equivalence calculations on complex lambda-expressions.

In passing we also mention another problem that did not occur in our initial, explicitly compositional, approach. When we label nodes with full semantic representations, there is no explicit information about how these representations were built up (or should be imagined to have been built up). Therefore, when a subtree is extracted out and the corresponding semantic abstraction takes place, it is not always uniquely defined which parts of a formula must be replaced by the unification variable. A simple example is the sentence "*Every man loves a man*". If one of the occurrences of "*man*" is abstracted out, it is not formally defined what replacements must be made in the formula at the sentence-node. Whether this kind of problem is statistically significant is not clear, however. (Every sentence-analysis process involves many different trees from the corpus, and most of these will not give rise to such confusions.)

Acknowledging these disadvantages, we nevertheless remain interested in the idea of annotating the nodes of the trees with full semantic representations rather than with compositional definitions. One reason is that we are ultimately pessimistic about the feasibility of that kind of annotation for large corpora of actual text. Another reason is, that full-representation-annotation is more easily compatible with the idea of *partial* annotations.

8.1.2 Not assuming surface compositionality: partial annotations

We now explore some situations where an initial, intuitively assigned annotation may be expected to be incomplete. One such situation is the phenomenon of non-standard quantifier scope. We illustrate this phenomenon with a well-known example sentence.

Suppose that the sentence "*Every man loves a woman*" occurs in a (larger) corpus with the following annotation (which in the context in which it was uttered was for some reason the most appropriate one for interpreting the utterance). The annotation gives the semantics of subconstituents only in so far as a meaning that can be locally established can also be straightforwardly recognized in the semantics of the total utterance.

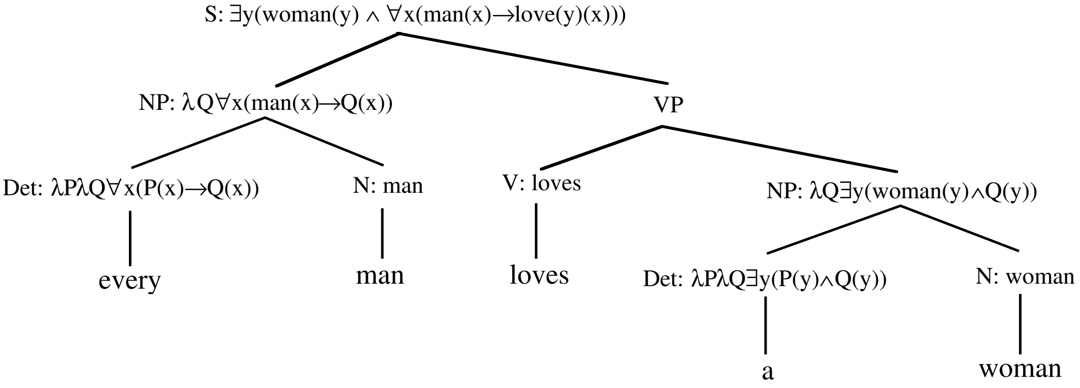


figure 8.11. Partial annotation for the sentence "*Every man loves a woman*"

The semantics of the whole utterance is known, as well as the semantics of the NPs *every man* and *a woman*. Also the interpretation of the verb *loves* is straightforward, but the semantics of the VP *loves a woman* is left unspecified, since the semantics that one would intuitively assign to this phrase does not occur as a sub-expression of the semantics of the complete utterance. Nevertheless, we do not want to exclude that subtrees headed by this VP node are employed as structural/semantic units in the parsing process. When, for this reason, the meaning of this VP is needed anyway, it can be arrived at by lambda-abstracting out the contribution of the VP's sister node from the semantics of the mother node.

To arrive at the meaning of the VP *loves a woman,* the system must abstract out the meaning of the NP *every man*, i.e. $\lambda Q\, \forall x(man(x) \rightarrow Q(x))$, from the meaning of the whole sentence, i.e. $\exists y(woman(y) \wedge \forall x(man(x) \rightarrow love(y)(x)))$. To make this possible, the meaning of the whole sentence must first be paraphrased as $\exists y(woman(y) \wedge \lambda Q\, \forall x(man(x) \rightarrow Q(x))\ (love(y)))$. Then, the NP-semantics $\lambda Q\, \forall x(man(x) \rightarrow Q(x))$ can be λ-abstracted straightforwardly out of the sentence-semantics, yielding $\lambda P \exists y(woman(y) \wedge P(love(y)))$.

Clearly, this treatment is reminiscent of the type-lifting rules of "Flexible Montague Grammar" (Hendriks, 1993). But since we are not writing a grammar but designing a performance model that works off semantically annotated trees, our treatment is a kind of mirror-image of the Flexible Montague Grammar approach. In our system, we start with the semantics of the full sentence, however complex or non-standard it may be. Like Flexible Montague Grammar, we then exploit the power of the lambda-calculus to enforce a compositional construction of that sentence semantics.

We expect that it is not realistic to assume that corpora are completely annotated. Partial annotations seem more appropriate because of phenomena like non-standard scope orders (like the example above), idioms, and discontinuous constituents. In a partially annotated corpus, meanings of nodes are not necessarily specified. For a sentence to count as analyzed and understood, what needs to be stored is (1) its syntactic structure and (2) its meaning.

Therefore, the only semantic feature on the tree that is necessarily specified is the one at the top; the meanings of sub-constituents may or may not be specified. But subtrees will only be useful for the semantic analysis of new input utterances, to the extent that they are either explicitly specified or formally derivable. The meaning of a constituent is derivable either by simple composition of the meanings of its subordinate constituents, or by abstracting out the contribution(s) of its sister node(s) in the semantics of its mother node.

An extreme case of a partial annotation may be provided by certain idioms. If we are given the sentence "*John kicked the bucket*" and are aware that this means that John died, we have the following structure in figure 8.12 with semantic annotation *die(john)*. If we add to this the knowledge that *John* is annotated with *john*, we can derive a meaning for the VP, but not further down. Thus, the analysis as it may occur in the corpus is:

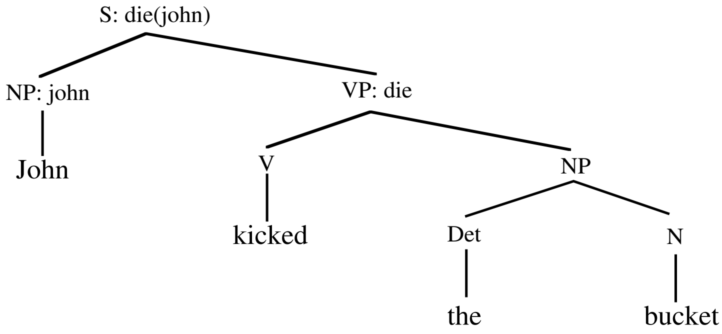


figure 8.12. Partial annotation for the sentence "*John kicked the bucket*"

This effectively means that semantically, but not syntactically, the whole idiomatic phrase *kicked the bucket* is treated as one word.

Another phenomenon that may give rise to partial annotations is provided by certain kinds of discontinuous constituency. Consider the following annotation of the sentence "*De Kerstman tuigde de kerstboom op*" (i.e. "*Santa Claus decorated the X-mas tree*").

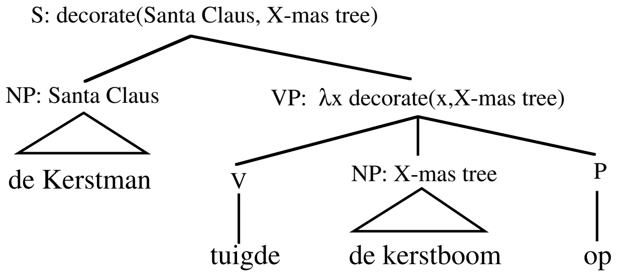


figure 8.13. Partial annotation for the sentence "*de Kerstman tuigde de kerstboom op*"

Again, the semantics at a node is specified only in so far as a meaning that can be locally established can also be straightforwardly recognized in the semantics of the total utterance. Thus, besides the meaning of the whole utterance, only the meanings of the NPs *de Kerstman* and *de kerstboom* and the meaning of the VP *tuigde de kerstboom op* are specified. The separate meanings of *tuigde* and *op* can neither be established locally, nor can they be arrived at by abstraction. However, the meaning of *tuigde* and *op* together can be established by decomposing the VP *tuigde de kerstboom op* into subtrees, i.e. by abstracting out the NP *de kerstboom*:

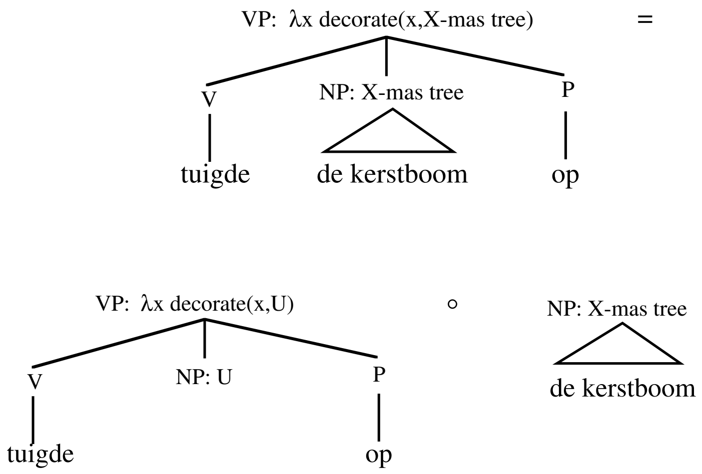


figure 8.14. Deriving structure and semantics for the discontinuous constituent "*tuigde op*"

Thus, the discontinuous constituent *tuigde op* can be used as a productive structural unit, also if it is separated by an NP.

In this semantic extension of DOP, any subtree can function as a productive unit, even if its semantics is not yet completely specified, provided its semantics can be calculated in the end by employing the principle of compositionality in one of two ways: (1) the meaning is constructed by simple composition of the constituents or (2) the meaning arrived at by abstracting out the contribution(s) of the sister node(s) from the semantics of the node directly governing it.

8.1.3 Summary: a performance model for semantic interpretation

By way of summary, we give the specifications of a performance model for semantic interpretation which we may call DOP6. This means that we instantiate the parameters of the DOP framework (sentence-analyses, sub-analyses, combination-operations, combination-probabilities):

(1) sentence-analyses

> partially semantically annotated phrase structure trees with specified top-node semantics

(2) sub-analyses

> subtrees

(3) combination-operations

> substitution/unification

(4) combination-probabilities

> the probability of substituting/unifying a certain subtree on another subtree: this is estimated by the ratio between the frequency of a subtree and the total frequency of subtrees with the same root-category.

The probability definitions of a derivation and parse in DOP6 are the same as in DOP1 and DOP3. That is, the probability of a derivation is equal to the product of the combination probabilities of its subtrees, and the probability of a parse tree is equal to the sum of the probabilities of its derivations. What is new in DOP6, is the probability of an interpretation of a string. An interpretation of a string is a formula which is logically equivalent to the semantic annotation of the top node of an analysis of this string. The probability of a string s with an interpretation I is the sum of the probabilities of the parse trees of string s with a top node annotated with a formula equivalent to I.

8.2 Future extensions: discourse and recency

In the prerequisites of our performance model in chapter 1, we assumed that the recency of previous language experiences does not influence the perceived analysis of a sentence. For the ATIS corpus, this assumption may be close to true, as this corpus merely contains distinct questions and imperatives without discourse structure. A sentence like "*Show me the flights from Boston to New York early in the morning*", can be properly disambiguated by only taking into account the frequencies of previous sub-analyses.

It has been recognized that not only the frequencies, but also the recencies of previous experiences bias the analysis of a new input (e.g. Kempen, 1994). A reason for this may be the impact of human memory limitations, but the impact of discourse structure may be even stronger. Consider the following two contiguous sentences:

I see a man with a bag
And I see a man with a telescope

It is clear that, due to the phenomenon of parallelism (cf. Prüst, 1992), the analysis of the first sentence creates a very strong bias in analyzing the second sentence. A performance model which wants to take this phenomenon into account, should not only register all previous analyses, but should also order them in time of occurrence. This implies that an adequate performance model should *update the corpus by every new language experience*..

If we limit ourselves for the moment to local discourse structure, how can we integrate the notion of recency into the DOP-framework, i.e. into the four-tuple (sentence-analyses, sub-analyses, combination-operations, combination-probabilities)? We strongly believe that the notion of recency can and should be integrated into the statistics of the performance model. That is, recency should be a function that adjusts the frequencies of occurrence of the analyses in such a way that the frequencies of more recently perceived analyses are adjusted upwards and those of less recently perceived analyses are adjusted downwards. The exact nature of this function is, however, unknown.

There is a further complication due to "global discourse structure" (cf. Scha & Polanyi, 1988). Consider the following sentences.

I see a man with a bag
    Look how old he is!
    He must be at least eighty
    He hits a dog with a stick
And there I see a man with a telescope

In this piece of discourse, the perceived analysis of the first sentence creates strong biases in the perception of the fourth sentence, and not so much in the perception of the sentences in between. Thus, the notion of recency needs to be integrated within a theory of discourse. How such an integration should be accomplished falls definitely outside the scope of this thesis.

8.3 Concluding remarks

In this thesis, we have shown how the frequencies of sub-analyses in a corpus of sentence-analyses can be used for predicting the perceived analysis of a new input sentence. We have shown the adequacy of the DOP-framework for syntactic language disambiguation. We developed the basics for an extension towards semantic interpretation with a note on discourse and recency. We are tempted to believe that by having a very large and fully analyzed language corpus, the problem of language performance can finally be solved. However, even then, our performance model will still have many limitations, some of which are worth mentioning here.

(1) We have only dealt with the perception of transcribed acoustic utterances. How does a performance model relate to the perception of naturally occurring speech? A first step has been made in (Bod & Scha, 1994), where a statistical model is presented which integrates sentence disambiguation and sentence prediction.

(2) Where is the place of language learning to be situated? With a fully analyzed corpus, only adult language behavior can be simulated. We believe that the problem of language acquisition is the acquisition of an initial corpus, in which non-linguistic input and pragmatics should play an important role.

(3) How does our notion of performance model relate to perception in other media, such as visual and musical perception? In (Bod & Scha, 1993) and (Scha & Bod, 1993), first steps have been made in investigating what is involved in extending DOP towards visual perception. A visual corpus can be seen as a collection of analyzed visual percepts, where a new visual percept can be analyzed by combining sub-analyses from the corpus. We conjecture that a generalization of our notion of performance model will finally stand as a general theory for human perception.

Bibliography

H. Alshawi, 1994. "Qualitative and Quantitative Models of Speech Translation", *The Balancing Act, Combining Symbolic and Statistical Approaches to Language, Proceedings of the Workshop*, ACL, New Mexico State University.

J. Balcazar, J. Diaz and J. Gabarro, 1988. *Structural Complexity*, Springer Verlag, Berlin.

M. van den Berg, R. Bod and R. Scha, 1994. "A Corpus-Based Approach to Semantic Interpretation", *Proceedings Ninth Amsterdam Colloquium*, Amsterdam.

E. Black, S. Abney, D. Flickenger, C. Gnadiec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski, 1991. "A Procedure for Quantitatively Comparing the Syntactic Coverage of English", *Proceedings DARPA Speech and Natural Language Workshop*, Pacific Grove, Morgan Kaufmann.

E. Black, J. Lafferty and S. Roukos, 1992. "Development and Evaluation of a Broad-Coverage Probabilistic Grammar of English-Language Computer Manuals", *Proceedings ACL'92*, Newark, Delaware.

E. Black, R. Garside and G. Leech, 1993. *Statistically-Driven Computer Grammars of English: The IBM/Lancaster Approach*, Rodopi: Amsterdam-Atlanta.

E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer and S. Roukos, 1993. "Towards History-Based Grammars: Using Richer Models for Probabilistic Parsing", *Proceedings ACL'93*, Columbus, Ohio.

R. Bod, 1992a. "Data Oriented Parsing (DOP)", *Proceedings COLING"92*, Nantes.

R. Bod, 1992b. "Mathematical Properties of the Data Oriented Parsing Model", *Preprints Third Meeting on Mathematics of Language*, Austin, Texas.

R. Bod, 1992c. "A Computational Model of Language Performance: Data Oriented Parsing", *Computational Linguistics in the Netherlands 1991*, Amsterdam.

R. Bod, 1993a. "Using an Annotated Corpus as a Stochastic Grammar", *Proceedings EACL'93*, Utrecht.

R. Bod, 1993b. "Monte Carlo Parsing", *Proceedings Third International Workshop on Parsing Technologies*, Tilburg/Durbuy.

R. Bod, 1993c. "Data Oriented Parsing as a General Framework for Stochastic Language Processing", in: K.Sikkel and A. Nijholt (eds.), *Parsing Natural Language*, TWLT6, Twente University.

R. Bod, 1993d. "Using an Annotated Language Corpus as a Virtual Stochastic Grammar", *Proceedings AAAI-93*, Menlo Park, Ca.

R. Bod, 1993e. "Applying Monte Carlo Techniques to Data Oriented Parsing", *Computational Linguistics in the Netherlands 1992*, Tilburg.

R. Bod, 1993f. "Combining Structural and Statistical Information from an Annotated Corpus", *Proceedings European Conference on Machine Learning (ECML"93)*, Vienna.

R. Bod, M. Dastani, H. Prüst, R. Scha and H. Zeevat, 1993. *HCI from a Discourse Perspective*, ESPRIT Basic Research Action P6296, Human Communication Research Centre, Edinburgh.

R. Bod and R. Scha, 1993. "Deriving Optimal Network Diagrams by means of Structural Information Theory", Bod, R. et al. (eds.), *HCI from a Discourse Perspective*, Edinburgh.

R. Bod, S. Krauwer and K. Sima'an, 1994. "CLASK: Combining Linguistic and Statistical Knowledge", ESF (European Science Foundation) Final Report 2a, Research Institute for Language and Speech, Utrecht University.

R. Bod and R. Scha, 1994. "Prediction and Disambiguation by means of Data-Oriented Parsing", *Proceedings Twente Workshop on Language Technology (TWLT8)*, Twente.

R. Bod, 1995. "The Problem of Computing the Most Probable Tree in Data-Oriented Parsing and Stochastic Tree Grammars", *Proceedings EACL'95*, Dublin.

T. Booth, 1969. "Probabilistic Representation of Formal Languages", *Tenth Annual IEEE Symposium on Switching and Automata Theory*.

T. Booth and R. Thompson, 1973. "Applying Probability Measures to Abstract Languages", *IEEE Transactions on Computers*, C-22(5).

T. Briscoe and J. Carroll, 1993. "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars", *Computational Linguistics* 19(1), 25-59.

T. Briscoe, 1994. "Prospects for Practical Parsing of Unrestricted Text: Robust Statistical Parsing Techniques", N. Oostdijk and P. de Haan (eds.), *Corpus-based Research into Language*, Rodopi, Amsterdam.

E. Brill, 1993. "Transformation-Based Error-Driven Parsing", *Proceedings Third International Workshop on Parsing Technologies*, Tilburg/Durbuy.

J. Carbonell, 1979. "Towards a Self-Extending Parser", *Proceedings ACL'79*.

J. Carroll and T. Briscoe, 1992. "Probabilistic Normalization and Unpacking of Packed Parse Forests for Unification-based Grammars", *Working Notes Probabilistic Approaches to Natural Language*, AAAI Fall Symposium Series, Cambridge, Ma.

E. Charniak, 1993. *Statistical Language Learning*, Cambridge (Ma): The MIT Press.

N. Chomsky, 1957. *Syntactic Structures*, The Hague: Mouton & Co.

N. Chomsky, 1965. *Aspects of the Theory of Syntax*, Cambridge (Ma): The MIT Press.

K. Church, 1988. "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text", *Proceedings ANLP'88*, Austin, Texas.

K. Church and R. Patil, 1983. *Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table*, MIT/LCS/TM-216.

K. Church and W. Gale, 1991. "A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams", *Computer Speech and Language* 5, 19-54.

K. Church and R. Mercer, 1993. "Introduction to the Special Issue on Computational Linguistics Using Large Corpora", *Computational Linguistics* 19(1), 1-24.

W. Cochran, 1963. *Sampling Techniques*, Wiley & Sons, New York (2nd edition).

A. Corazza, R. Gretter and G. Satta, 1991. "Stochastic Context-Free Grammars for Island-Driven Probabilistic Parsing", *Proceedings Second International Workshop on Parsing Technologies*, Cancun, Mexico.

(D)ARPA Proceedings on Speech and Natural Language, 1991, 1992, 1993, 1994. Morgan Kaufmann, San Mateo. Ca.

W. Deming, 1966. *Some Theory of Sampling*, Dover Publications, New York.

J. Earley, 1970. "An Efficient Context-free Parsing Algorithm", *Communications ACM*, 13(2), 94-102.

W. Feller, 1950. *An Introduction to Probability Theory and its Applications*, Wiley, New York.

G. Fenk-Oczlon, 1989. "Word frequency and word order in freezes", *Linguistics* 27, 517-556.

K. Fu, 1974. "Syntactic methods in pattern recognition", *Mathematics in Science and Engineering, 112*, Academic Press.

K. Fu, 1982. *Syntactic Pattern Recognition and Applications*, Prentice-Hall.

T. Fujisaki, 1984. "An Approach to Stochastic Parsing", *Proceedings COLING-84*.

T. Fujisaki, F. Jelinek, J. Cocke, E. Black and T. Nishino, 1989. "A Probabilistic Method for Sentence Disambiguation", *Proceedings 1st Int. Workshop on Parsing Technologies*, Pittsburgh.

W. Gale and K. Church, 1994. "What is wrong with adding one?", N. Oostdijk and P. de Haan (eds.), *Corpus-based Research into Language*, Rodopi, Amsterdam.

R. Garside, G. Leech and G. Sampson, 1987. *The Computational Analysis of English: A Corpus-Based Approach*, Longman.

I. Good, 1953. "The Population Frequencies of Species and the Estimation of Population Parameters", *Biometrika* 40, 237-264.

S. Graham, M. Harrison and W. Ruzzo, 1980. "An improved context-free recognizer", *ACM Transactions on Programming Languages and Systems*, 2(3): 415-462.

R. Grishman, C. Macleod and J. Sterling, 1992. "Evaluating Parsing Strategies Using Standardized Parse Files", *Proceedings ANLP'92*, Trento.

H. van Halteren and N. Oostdijk, 1988. "Using an Analyzed Corpus as a Linguistic Database", J. Roper (ed.) *Computers in Literary and Linguistic Computing*, Champion Slatkine, Paris.

J. Hammersley and D. Handscomb, 1964. *Monte Carlo Methods*, Chapman and Hall, London.

P. Harrison, S. Abney, E. Black, D. Flickenger, C. Gnadiec, R. Grishman, D. Hindle, R. Ingria, M. Marcus, B. Santorini and T. Strzalkowski, 1991. "Evaluating Syntax Performance of Parser/Grammars", *Proceedings of the Natural Language Processing Systems Evaluation Workshop*, Berkeley.

I. Hasher and W. Chromiak, 1977. "The processing of frequency information: an automatic mechanism?", *Journal of Verbal Learning and Verbal Behavior* 16, 173-184.

I. Hasher and R. Zacks, 1984. "Automatic Processing of Fundamental Information: the case of frequency of occurrence", *American Psychologist* 39, 1372-1388.

C. Hemphill, J. Godfrey and G. Doddington, 1990. "The ATIS spoken language systems pilot corpus". *Proceedings DARPA Speech and Natural Language Workshop*, Hidden Valley, Morgan Kaufmann.

H. Hendriks, 1993. *Studied Flexibility, Categories and Types in Syntax and Semantics*, ILLC Dissertation Series 1993-5.

D. Hindle and M. Rooth, 1993. "Structural Ambiguity and Lexical Relations", *Computational Linguistics*, 19(1), 103-120.

L. Jacoby and L. Brooks, 1984. "Nonanalytic Cognition: Memory, Perception and Concept Learning", G. Bower (ed.), *Psychology of Learning and Motivation* (Vol. 18, 1-47), San Diego: Academic Press.

H. Jeffreys, 1948. *Theory of Probability*, Second Edition, Section 3.23, Oxford, Clarendon Press.

F. Jelinek, 1985. "The Development of an Experimental Discrete Dictation Recognizer", *IEEE'85* (Invited Paper).

F. Jelinek and R. Mercer, 1985. "Probability Distribution Estimation from Sparse Data", *IBM Technical Disclosure Bulletin* 28, 2591-2594.

F. Jelinek, J. Lafferty and R. Mercer, 1990. *Basic Methods of Probabilistic Context Free Grammars*, Technical Report IBM RC 16374 (#72684), Yorktown Heights.

W. Johnson, 1932. Appendix (edited by R. Braithwaite) to "Probability Deductive and Inductive Problems", *Mind*, 41, 421-423.

A. Joshi, 1987. "Introduction to Tree-Adjoining Grammar", A. Manaster Ramer (ed.), *The Mathematics of Language*, J. Benjamins.

A. Joshi and Y. Schabes, 1991. "Tree-Adjoining Grammars and Lexicalized Grammars", M. Nivat (ed.), *Definability and Recognizability of Sets of Trees*, Elsevier.

A. Joshi, K. Vijay-Shanker and D. Weir, 1991. "The Convergence of Mildly Context-Sensitive Grammar Formalisms", Sells et al. (eds.), *Foundational Issues in NLP*, MIT Press.

C. Juliano and M. Tanenhaus, 1993. "Contingent Frequency Effects in Syntactic Ambiguity Resolution", *Fifteenth Annual Conference of the Cognitive Science Society*, 593-598, Hillsdale, NJ.

F. Karlsson, A. Voutilainen, J. Heikkila and A. Anttila, 1995. *Constraint Grammar, A Language-Independent System for Parsing Unrestricted Text*, Mouton de Gruyter: Berlin.

S. Katz, 1987. "Estimation of probabilities from sparse data for the language model component of a speech recognizer", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35, 400-401.

D. Kausler and J. Puckett, 1980. "Frequency Judgments and Correlated Cognitive Abilities in Young and Elderly Adults", *Journal of Gerontology* 35, 376-382.

M. Kay, 1980. *Algorithmic Schemata and Data Structures in Syntactic Processing*. Report CSL-80-12, Xerox PARC, Palo Alto, Ca.

G. Kempen, 1994. "Computational Models of Syntactic Processing in Human Language Comprehension", in A. Dijkstra & K. de Smedt (eds.), *Computational Psycholinguistics: Symbolic and subsymbolic models of language processing*. Taylor & Francis, London.

W. Levelt, 1974. *Formal Grammars in Linguistics and Psycholinguistics (vol.I)*, Mouton, The Hague.

M. Liberman, 1991. "The Trend towards Statistical Models in Natural Language Processing", in E. Klein and F. Veltman (eds.), *Natural Language and Speech*, Springer Verlag, Berlin.

M. Liberman and Y. Schabes, 1993. *Statistical Methods in Natural Language Processing*, Tutorial notes, EACL-93, Utrecht.

Longman Dictionary of the English Language, 1988, Longman, London.

M. MacDonald, N. Pearlmutter and M. Seidenberg, 1994. "Lexical Nature of Syntactic Ambiguity Resolution", *Psychological Review* 101, 676-703.

D. Magerman and M. Marcus, 1991. "Pearl: A Probabilistic Chart Parser", *Proceedings EACL'91*, Berlin.

D. Magerman and C. Weir, 1992. "Efficiency, Robustness and Accuracy in Picky Chart Parsing", *Proceedings ACL'92*, Newark, Delaware.

D. Magerman, 1995. "Statistical Decision-Tree Models for Parsing", *Proceedings ACL'95*, Cambridge, Massachusetts.

M. Marcus, B. Santorini and M. Marcinkiewicz, 1993. "Building a Large Annotated Corpus of English: the Penn Treebank", *Computational Linguistics* 19(2).

W. Martin, K. Church and R. Patil, 1987. "Preliminary Analysis of a Breadth-first Parsing Algorithm: Theoretical and Experimental Results", in: L. Bolc (ed.), *Natural Language Parsing Systems*, Springer Verlag, Berlin.

H. Meyer (ed.), 1956. *Symposium on Monte Carlo Methods*, Wiley, New York.

D. Mitchell, F. Cuetos and M. Corley, 1992. "Statistical versus Linguistic Determinants of Parsing Bias: Cross-linguistic Evidence", *Fifth Annual CUNY Conference on Human Sentence Processing*, New York.

A. Nadas, 1985. "On Turing's formula for word probabilities", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33, 1414-1416.

J. Nicol and M. Pickering, 1993. "Processing Syntactically Ambiguous Sentences: Evidence from Semantic Priming", *Journal of Psycholinguistic Research*.

N. Pearlmutter and M. MacDonald, 1992. "Plausibility and Syntactic Ambiguity Resolution", *Proceedings 14th Annual Conf. of the Cognitive Society*.

F. Pereira and Y. Schabes, 1992. "Inside-Outside Reestimation from Partially Bracketed Corpora", *Proceedings ACL'92*, Newark.

P. Price, 1994. "Combining Linguistic with Statistical Methods in Automatic Speech Understanding", *The Balancing Act, Combining Symbolic and Statistical Approaches to Language, Proceedings of the Workshop*, ACL, New Mexico State University.

H. Prüst, 1992. *On Discourse Structuring, VP Anaphora and Gapping*, PhD Thesis, Dept. of Computational Linguistics, University of Amsterdam.

P. Resnik, 1992. "Probabilistic Tree-Adjoining Grammar as a Framework for Statistical Natural Language Processing", *Proceedings COLING'92*, Nantes.

G. Sampson, 1986. "A Stochastic Approach to Parsing", *Proceedings COLING'86*, Bonn.

G. Sampson, 1987. "Evidence against the 'Grammatical/Ungrammatical' Distinction", W. Meijs (ed.), *Corpus Linguistics and Beyond*, Rodopi, Amsterdam.

B. Santorini, 1990. *Part-of-Speech Tagging Guidelines for the Penn Treebank Project*, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia.

B. Santorini, 1991. *Bracketing Guidelines for the Penn Treebank Project*, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia.

R. Scha and L. Polanyi, 1988. "An Augmented Context-free Grammar for Discourse", *Proceedings COLING'88*, Budapest.

R. Scha, 1990. "Taaltheorie en Taaltechnologie; Competence en Performance", in Q.A.M. de Kort and G.L.J. Leerdam (eds.), *Computertoepassingen in de Neerlandistiek*, Almere: Landelijke Vereniging van Neerlandici (LVVN-jaarboek).

R. Scha, 1992. "Virtuele Grammatica's en Creatieve Algoritmen", *Gramma/TTT* 1(1).

R. Scha and R. Bod, 1993. "Computationele Esthetica", *Informatie en Informatiebeleid* 11(1).

Y. Schabes, 1991. "Polynomial Time and Space Shift-Reduce Parsing of Arbitrary Context-free Grammars", *Proceedings ACL'91*, Berkeley, Ca.

Y. Schabes, 1992. "Stochastic Lexicalized Tree-Adjoining Grammars", *Proceedings COLING'92*, Nantes.

Y. Schabes, M. Roth and R. Osborne, 1993. "Parsing the Wall Street Journal with the Inside-Outside Algorithm", *Proceedings EACL'93*, Utrecht.

Y. Schabes and R. Waters, 1993. "Stochastic Lexicalized Context Free Grammars", *Proceedings Third International Workshop on Parsing Technologies*, Tilburg/Durbuy.

K. Sima'an, R. Bod, S. Krauwer and R. Scha, 1994. "Efficient Disambiguation by means of Stochastic Tree Substitution Grammars", *Proceedings International Conference on New Methods in Language Processing*, UMIST, Manchester.

R. Simmons and Y. Yu, 1992. "The Acquisition and Use of Context-Dependent Grammars for English", *Computational Linguistics* 18(4), 391-418.

R. Smith, 1973. *Probabilistic Performance Models of Language*, Mouton, The Hague.

P. Suppes, 1970. "Probabilistic Grammars for Natural Languages", *Synthese* 22.

A. Viterbi, 1967. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. Information Theory*, IT-13, 260-269.

R. Weischedel, M. Meteer, R, Schwarz, L. Ramshaw and J. Palmucci, 1993. "Coping with Ambiguity and Unknown Words through Probabilistic Models", *Computational Linguistics*, 19(2), 359-382.

C. Wetherell, 1980. "Probabilistic Languages: A Review and Some Open Questions", *Computing Surveys*, 12(4).

T. Winograd, 1983. *Language as a Cognitive Process. Volume I: syntax*. Reading (Mass.).

J. Wright, E. Wrigley and R. Sharman, 1991. "Adaptive Probabilistic Generalized LR Parsing", *Proceedings Second International Workshop on Parsing Technologies*, Cancun, Mexico.

D. Younger, 1967. "Recognition and Parsing of Context-free Languages in time n^3", *Information and Control*, 10(2), 189-208.

G. Zipf, 1935. *The Psycho-Biology of Language*, Houghton Mifflin.

Samenvatting

Een van de grootste problemen in de hedendaagse (computationele) linguistiek is de ambiguiteit van natuurlijke taal: zodra een grammatica een non-triviaal gedeelte van een natuurlijke taal karakteriseert, krijgt vrijwel elke zin van enige lengte zeer veel verschillende syntactische analyses (en bijbehorende semantische interpretaties). Dit is problematisch omdat het merendeel van deze interpretaties door een menselijke taalgebruiker helemaal niet als mogelijk wordt waargenomen. Een taalgebruiker neemt gewoonlijk niet meer dan een of twee interpretaties waar. In zekere zin is dit probleem niet verrassend: het is een onmiddellijk gevolg van het feit dat formele linguistische modellen direct aansluiten bij Chomsky's notie van een "competence grammatica". Chomsky heeft altijd nadrukkelijk onderscheid gemaakt tussen de "competence" van een taalgebruiker en diens "performance". De competence is de taalkennis waarover de taalgebruiker in principe beschikt; de performance is het psychologische proces dat die kennis gebruikt voor feitelijke taalproductie en -perceptie. De voorkeuren die taalgebruikers hebben in het geval van meerduidige interpretaties behoren typisch tot het domein dat in een Chomskyaanse optiek tot de performance zou worden gerekend.

Het doel van dit proefschrift is te laten zien hoe een statistische verrijking van een linguistisch "competence model" de invoer-uitvoer eigenschappen van menselijke taalwaarneming kan verantwoorden. Het resulterende "performance model" zal in staat zijn om uit alle mogelijke interpretaties van een zin de daadwerkelijk waargenomen interpretatie te kiezen. Dat zo'n performance model statistisch van aard dient te zijn, laten we zien aan de hand van een aantal resultaten uit de psycholinguistiek (§1.1). Dit leidt tot de stelling dat een taalgebruiker neigt naar het waarnemen van de meest waarschijnlijke interpretatie van een invoer-zin op basis van voorkomens van eerder waargenomen interpretaties. Deze stelling brengt ons tot de notie van een performance model dat een geanalyseerd taalcorpus gebruikt voor het berekenen van de meest waarschijnlijke analyse van een nieuwe invoer-zin middels combinaties van deel-analyses uit het corpus.

In § 1.3 wordt onze notie van performance model in een breder kader geplaatst, het zgn. Data-Oriented Parsing (DOP) framework. In dit framework onderscheiden we vier parameters: (1) definitie van de zinsanalyses in het corpus, (2) definitie van de deel-analyses, (3) definitie van de combinatie-operaties tussen deel-analyses, en (4) definitie van de combinatie-kansen van deel-analyses. Middels dit framework kan een groot aantal performance modellen worden geinstantieerd. Wij leggen ons de beperking op dat we alleen die parameter-instellingen kiezen waarvan de resulterende performance modellen ook daadwerkelijk computationeel kunnen worden getest met beschikbare corpora.

Als test-procedure wordt de zogenaamde "blind testing methode" voorgesteld (§1.4). Deze methode schrijft voor dat een geanalyseerd taalcorpus "at random" wordt verdeeld in een zogenaamde training-set en een test-set. Alleen de analyses van de training-set zinnen mogen worden gebruikt om het systeem te trainen (d.i. het schatten van de combinatie-kansen van de deel-analyses). De zinnen uit de test-set worden vervolgens automatisch geanalyseerd ("geparseerd") en vergeleken met de analyses uit de test-set. De mate waarin de meest waarschijnlijke analyses overeenkomen met de analyses uit de test-set geeft de parseer-nauwkeurigheid van het systeem aan.

In het tweede hoofdstuk van dit proefschrift wordt een eerste realisatie van een performance model binnen het DOP-framework uitgewerkt, die we DOP1 noemen. DOP1 gebruikt "deelbomen" uit het corpus als deel-analyses en "compositie" als combinatie-operatie tussen deel-analyses. Voor het schatten van de compositie-kansen van deelbomen wordt gebruik gemaakt van twee assumpties: (1) de deelbomen zijn statistisch onafhankelijk, en (2) de verzameling deelbomen representeert de totale populatie van deelbomen. Het belangrijkste kenmerk van DOP1 is het onderscheid tussen de kans van een derivatie (of afleiding) van een zin en de kans van een parseerboom (of analyse) van een zin. De kans van een derivatie is gelijk aan het product van de compositie-kansen van de gebruikte deelbomen, terwijl de kans van een parseerboom gelijk is aan de som van de kansen van de verschillende derivaties die deze boom genereren.

Hoofdstuk 3 introduceert een Formele Talen Theorie van Stochastische Grammatica's waarbinnen verschillende stochastische taalmodellen kunnen worden gearticuleerd en vergeleken. We beschrijven DOP1 als een projectie van een boom-verzameling in een "Stochastic Tree-Substitution Grammar" (STSG), en vergelijken op formele wijze STSG met de bekende "Stochastic Context-Free Grammar" (SCFG). Een belangrijk resultaat van deze vergelijking is dat SCFGs stochastisch zwakker zijn dan STSGs: de verzameling van stochastische boom-talen gegenereerd door SCFGs is een deelverzameling van de verzameling van stochastische boom-talen gegenereerd door STSGs. We vergelijken STSG ook met twee andere stochastische grammatica's die in de literatuur zijn voorgesteld om de statistische context-ongevoeligheid van SCFGs te boven te komen: "Stochastic History-Based Grammar" en "Stochastic Tree-Adjoining Grammar". Het blijkt dat deze stochastische grammatica's niet alle statistische afhankelijkheden kunnen beschrijven die kunnen worden beschreven door STSG.

In hoofdstuk 4 wordt het probleem van het berekenen van de meest waarschijnlijke parseerboom van een zin in DOP1 behandeld. We maken onderscheid tussen parseren en disambigueren, en tonen aan dat het probleem niet zozeer ligt in de constructie van een zogenaamd "parse-forest" voor een zin, maar in de selectie van de meest waarschijnlijke parseerboom uit dit forest. We laten zien dat een Viterbi-optimalisatie niet toepasbaar is voor het vinden van de meest waarschijnlijke parseerboom. We stellen een iteratieve Monte Carlo procedure voor die de meest waarschijnlijke analyse kan schatten met een fout die willekeurig klein kan worden gemaakt in polynomiale tijd. Tenslotte, gaan we in op enige psychologisch interessante eigenschappen van Monte Carlo disambiguering.

In hoofdstuk 5 testen we de verdiensten van DOP1 als performance model voor syntactische disambiguering. Experimenten op part-of-speech sequenties van het "Air-Travel Information System" (ATIS) corpus resulteren in 96% parseer-nauwkeurigheid, hetgeen substantieel hoger is dan de parseer-nauwkeurigheid van andere systemen. Als de grootte van de corpus-deelbomen wordt beperkt, neemt de parseer-nauwkeurigheid af tot 52% bij een deelboom-diepte van één. Het blijkt dat predicties die zijn gebaseerd op de meest waarschijnlijke parseerboom zeer veel accurater zijn dan predicties gebaseerd op de parseerboom die wordt gegenereerd door de meest waarschijnlijke derivatie. We testen ook hoeveel de eliminatie van eenmaal-voorkomende deelbomen de parseer-nauwkeurigheid beinvloedt, en in welke mate de grootte van de training-set de nauwkeurigheid beinvloedt. Tenslotte stellen we vast dat DOP1 100% nauwkeurigheid haalt als de training-set en test-set worden samengevoegd, terwijl SCFG in dat geval slechts 59% nauwkeurigheid haalt.

Hoofdstuk 6 begint met een onderzoek naar wat nodig is om DOP1 uit te breiden zodanig dat zinnen met onbekende woorden kunnen worden geparseerd. Het model DOP2 wordt voorgesteld als een zeer eenvoudige extensie van DOP1: onbekende woorden worden gelabeld met alle lexicale categorieen, waarna DOP1 wordt gebruikt voor het berekenen van de meest waarschijnlijke parseerboom. Experimenten met DOP2 op woord-sequenties uit het ATIS corpus, laten echter een teleurstellende parseer-nauwkeurigheid van 63% zien. Een kwalitatieve evaluatie van de testzinnen met onbekende woorden wijst uit dat DOP2 een afwijking heeft naar het gebruik van kleinere deelbomen. De evaluatie van testzinnen met alleen bekende woorden leidt tot de notie van "onbekende-categorie woord": een meerduidig woord dat in de training-set voorkomt, maar met een andere categorie dan nodig is om de test-zin met dit woord correct te parseren. Het blijkt dat DOP2 inadequaat is voor het resolveren van onbekende-categorie woorden.

De resultaten van hoofdstuk 6 leiden tot het performance model DOP3. Het belangrijkste inzicht van DOP3 is dat de notie van "onbekende deelboom" de problemen met zowel onbekende woorden als onbekende-categorie woorden zou kunnen oplossen. Teneinde met onbekende deelbomen om te kunnen gaan, beperken we ons tot deelbomen waarvan de onbekendheid afhangt van onbekende terminale symbolen. Het voornaamste probleem blijkt dan de schatting te zijn van de frequenties van onbekende deelbomen. Om dit probleem op te lossen, laten we de assumptie vallen dat alle deelbomen zijn waargenomen, en behandelen we een corpus als een sample van een grotere populatie. We gebruiken de Good-Turing methode voor het schatten van de populatie-kansen van zowel onbekende als bekende deelbomen. Dit leidt tot de definitie van het performance model DOP3. Experimenten tonen aan dat DOP3 redelijk succesvol zinnen met onbekende (-categorie) woorden kan parseren en disambigueren: DOP3 behaalt 83% parseer-nauwkeurigheid bij een deelboom-diepte ≤ 3. Bovendien lijdt DOP3 niet meer onder de afwijking in de richting van kleinere deelbomen als in DOP2. De Good-Turing methode wordt vergeleken met de zgn. Add-k methode, resulterend in DOP4. Het blijkt dat DOP4 een slechtere parseer-nauwkeurigheid oplevert dan DOP3. Om de best mogelijke resultaten te verkrijgen, wordt DOP3 uiteindelijk verrijkt met een extern woordenboek (Longman), hetgeen het hybride model DOP5 oplevert, dat een nauwkeurigheid bereikt van 92% (voor deelboom-diepte ≤ 3).

Het laatste hoofdstuk onderzoekt wat nodig is om een syntactisch geanalyseerd taalcorpus uit te breiden met semantische interpretaties. We laten zien dat als we "oppervlakte-compositionaliteit" aannemen, het syntactische annotatie probleem kan worden gegeneraliseerd naar het probleem van semantische annotatie. Voor het analyseren van echte tekst lijken partiele annotaties echter realistischer. We stellen het performance model DOP6 voor, waarin elke deelboom uit het corpus kan functioneren als productieve eenheid, ook als de semantiek ervan nog niet volledig is gespecificeerd, mits de semantiek uiteindelijk kan worden berekend middels het compositionaliteits-principe op een van de volgende twee manieren: (1) de betekenis kan worden geconstrueerd middels eenvoudige compositie van de constituenten of (2) de betekenis kan worden afgeleid middels het abstraheren van de contributie(s) van de zuster kno(o)p(en) uit de semantiek van de direct bovenliggende knoop. Het belangrijkste statistische kenmerk van DOP6 is de waarschijnlijkheid van een interpretatie I van een zin als de som van de kansen van alle parseerbomen die een top-knoop semantiek hebben die logisch equivalent is aan I. Tenslotte, als een belofte voor toekomstig onderzoek, behandelen we de invloed van discourse-structuur en recentheid op de analyse van een invoer-zin.

Titles in the ILLC Dissertation Series:

Transsentential Meditations; Ups and downs in dynamic semantics
Paul Dekker
ILLC Dissertation Series 1993-1

Resource Bounded Reductions
Harry Buhrman
ILLC Dissertation Series 1993-2

Efficient Metamathematics
Rineke Verbrugge
ILLC Dissertation Series 1993-3

Extending Modal Logic
Maarten de Rijke
ILLC Dissertation Series 1993-4

Studied Flexibility
Herman Hendriks
ILLC Dissertation Series 1993-5

Aspects of Algorithms and Complexity
John Tromp
ILLC Dissertation Series 1993-6

The Noble Art of Linear Decorating
Harold Schellinx
ILLC Dissertation Series 1994-1

Generating Uniform User-Interfaces for Interactive Programming Environments
Jan Willem Cornelis Koorn
ILLC Dissertation Series 1994-2

Process Theory and Equation Solving
Nicoline Johanna Drost
ILLC Dissertation Series 1994-3

Calculi for Constructive Communication, a Study of the Dynamics of Partial States
Jan Jaspars
ILLC Dissertation Series 1994-4

Executable Language Definitions, Case Studies and Origin Tracking Techniques
Arie van Deursen

ILLC Dissertation Series 1994-5

Chapters on Bounded Arithmetic & on Provability Logic
Domenico Zambella
ILLC Dissertation Series 1994-6

Adventures in Diagonalizable Algebras
V. Yu. Shavrukov
ILLC Dissertation Series 1994-7

Learnable Classes of Categorial Grammars
Makoto Kanazawa
ILLC Dissertation Series 1994-8

Clocks, Trees and Stars in Process Theory
Wan Fokkink
ILLC Dissertation Series 1994-9

Logics for Agents with Bounded Rationality
Zhisheng Huang
ILLC Dissertation Series 1994-10

On Modular Algebraic Protocol Specification
Jacob Brunekreef
ILLC Dissertation Series 1995-1

Investigating Bounded Contraction
Andreja Prijatelj
ILLC Dissertation Series 1995-2

Algebraic Relativization and Arrow Logic
Maarten Marx
ILLC Dissertation Series 1995-3

Studies on the Formal Semantics of Pictures
Dejuan Wang
ILLC Dissertation Series 1995-4

Generation of Program Analysis Tools
Frank Tip
ILLC Dissertation Series 1995-5

Verification Techniques for Elementary Data Types and Retransmission Protocols
Jos van Wamel
ILLC Dissertation Series 1995-6

Transformation and Analysis of (Constraint) Logic Programs
Sandro Etalle
ILLC Dissertation Series 1995-7

Frames and Labels. A Modal Analysis of Categorial Inference
Natasha Kurtonina
ILLC Dissertation Series 1995-8

Tools for PSF
G.J. Veltink
ILLC Dissertation Series 1995-9

(to be announced)
Giovanna Cepparello
ILLC Dissertation Series 1995-10

Instantial Logic. An Investigation into Reasoning with Instances
W.P.M. Meyer Viol
ILLC Dissertation Series 1995-11

Taming Logics
Szabolcs Mikulas
ILLC Dissertation Series 1995-12

Metalogics for Logic Programming
Marianne Kalsbeek
ILLC Dissertation Series 1995-13

Enriching Linguistics with Statistics: Performance Models of Natural Language
Rens Bod
ILLC Dissertation Series 1995-14

Computational Pitfalls in Tractable Grammatical Formalisms
Marten Henrik Trautwein
ILLC Dissertation Series 1995-15

The Solution Sets of Local Search Problems
Sophie Titia Fischer
ILLC Dissertation Series 1995-16