Back-off as Parameter Estimation for DOP models

Luciano Grüdtner Buratto
Master of Logic Thesis
Institute for Logic, Language and Computation
Universiteit van Amsterdam

November, 18th 2002

Contents

Acknowledgements

This thesis would not be possible without the help of several people. I am very grateful to my supervisor Khalil Sima'an for his endurance during the moments I thought the whole work was going astray. The little I know about Natural Language Processing I owe to him.

I would also like to thank my friends who made my days in Amsterdam much happier: Bernadette Martinez, for being a good friend and for understanding my Portuñol (Portuguese mixed up with Spanish); Börkur Sigurbjörnsson, whose unpronounceable name hides a person with interesting political views and some strange drinking habits (never try 'Black Death'); Irwin Lim, for the discussions in Nunspeet that were decisive in my understanding of key logic concepts, for introducing me to the Python programming language and for cooking wonderfully; Ravi Rajani, for being a Brit that thinks is Dutch and for having the finest musical taste (João Gilberto, Tom Jobim) and Steffen Bauer, for teaching me what not to do when going out in Amsterdam.

I am also indebted with Nuffic for supporting my stay in the Netherlands.

Finally, I thank my parents, Paulo and Julieta, who, from the small town of Urubici, in Southern Brazil, always believed in and cared for me; my brother, Gabriel, whose friendship is so important to me, and Noeli, whose love and company, even though long-distance, brought me comfort and strength during this long year.

Abstract

Data-Oriented Parsing (DOP) is a probabilistic performance approach to parsing natural language. Several DOP models have been proposed since it was introduced by Scha (1990), achieving promising results. One important feature of these models is the probability estimation procedure. Two major estimators have been put forward: Bod (1993) uses a relative frequency estimator; Bonnema (1999) adds a rescaling factor to correct for tree size effects. Both estimators, however, present biases. Moreover, Bod's estimator has been shown to be inconsistent (Johnson, 2002), meaning that the probability estimates hypothesized by the model do not approach the true probabilities that generated the data as the sample size grows. In this thesis, we implement a new estimation procedure that tackles the shortcomings of the two previous methods. The main idea is to treat derivation events not as disjoint, but as interrelated in a hierarchical cascade of parse tree derivations. We show that this new estimator – called the Back-Off DOP (BO-DOP) estimator – outperforms both previous models. We tested it on the OVIS treebank, a Dutch language, speech-based system, and report error reductions of up to 11.4% and 15% when compared to, respectively, Bod's and Bonnema's estimators.

Chapter 1

Data-Oriented Parsing: virtues and limitations

Parsing natural language is a computationally difficult task due to the inherent ambiguity of natural language grammars. Given a sentence, the number of possible structure assignments can grow exponentially with sentence length (Church and Patil, 1982). Most of these analyses are not perceived by humans, since extra-linguistic information helps discarding implausible interpretations and selecting the most likely one. For computers, however, disambiguation is a much harder task, since contextual information is rarely available.

Several approaches have been proposed to tackle automatic natural language parsing. The one described in this thesis is called *Data-Oriented Parsing* (**DOP**). It defines an entire class of probabilistic performance models whose common underlying intuition is the following: *<u>all</u> patterns of past linguistic experience and their relative frequencies can influence the processing of new input and therefore should be taken into account.*

DOP is a class of *performance* models because it describes language by looking at the actual utterances that native speakers produce. This is in contrast to *competence models* (Chomsky, 1957), which view language as a system of principles and rules that can be unveiled by a linguist's intuition. The main goal of a competence grammar is to describe which sentences and analyses are to be considered *grammatical*. Given one grammatical sentence and two possible analyses, little information is provided as to which structure is to be preferred. When it comes to performance models and language technology, however, information about language use is essential. By looking at large collections of annotated analyses, performance models can capture such preferences.

DOP is a class of *probabilistic* models because it uses frequency information to rank analyses and choose the one yielding the expected interpretation. Frequency information has been shown to play an important role in cognitive processing (Chater, 1996; Chater et al., 1998) and has been successfully applied in language technology tasks, such as *speech recognition* (Jelinek, 1999).

Contrary to other probabilistic performance approaches, DOP makes no assumptions regarding linguistic structures. Lexicalized Tree-Adjoining Grammars (LTAG), for instance, define beforehand which structures are to be considered relevant; only those structures are assigned probabilities (Srinivas and Joshi, 1999). In DOP, the statistical units are made up of all fragments that can be extracted from utterance-analyses. As a result, DOP models are able to capture more long-distance, lexical and structural dependencies. Unlike head-lexicalized models (Collins, 1997), which restrict the relevant statistical dependencies to pairs of head-words, DOP models are able to capture dependencies between more than two words and between unlexicalized structures, yielding more fine-grained probability distributions. Indeed, DOP models stand at the top of a subsumption lattice of probabilistic models having *Stochastic Context-Free Grammars* (SCFG) at the bottom, as the least discriminating models, and DOP models at the top (Carroll and Weir, 2000).

The Data-Oriented Parsing approach was introduced by Scha (1990) as a response to both competence grammars and linguistically motivated SCFGs. Even though probabilistic in nature, the latter are too simplistic, since they cannot identify frequently occurring phrases (e.g. LET ME KNOW), expressions (e.g. TAKE ADVANTAGE OF) or figures of speech (e.g. HOT AS HELL).

The probability estimation procedure is an important component in the definition of a particular DOP model. Parameter estimation in DOP models is the focus of this thesis. We show the shortcomings of previous methods and propose a new estimator. In the remaining of this chapter we formally define DOP models, describe instantiations of such models and explain some of their limitations, focusing on the estimation procedure.

1.1 DOP1

1.1.1 Example and definitions

A particular DOP model is completely defined if it specifies:

- a representation of utterance-analyses (e.g. phrase-structure)
- a fragment extraction procedure (e.g. subtree generation)
- a combining operation (e.g. substitution)
- an estimation procedure (e.g. relative frequency)

The first instantiation of a DOP model is due to Bod (1992) and is referred to as **DOP1**. Since then, several models following the same underlying principles have been proposed, hoping at improving previous versions by adding more detailed linguistic and probabilistic information. One first class changed the representation of utterance-analyses, enriching them with feature information (LFG-DOP: Bod and Kaplan, 1998) or semantic information (Bonnema et al., 1997). A second class changed the extraction procedure and the combining operation (TIG-DOP: Hoogweg, 2000; Tree-grams[1]: Sima'an, 2001b). Finally, the estimation procedure has been changed (Bonnema, Buying, and Scha, 1999).

[1]Tree-grams also changed the utterance-analyses, enriching them with head information.

In this section we describe the DOP1 model, specifying how it instantiates the four requirements laid out above, and define some general concepts that will be used throughout this thesis.

In DOP1, utterance-analyses are represented as phrase-structure trees. A *tree* is a connected graph without cycles. A tree whose frontier consists of only lexical items is called a *constituent*. A *corpus* or *treebank* is a collection of constituents. The term collection is used to emphasize that $\tau_i = \tau_j$ may hold for $i \neq j$. Assume a corpus consisting of only two trees, as shown in Figure 1.1.

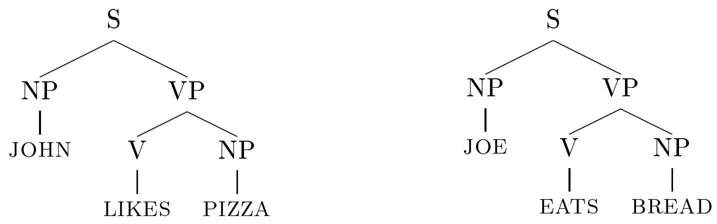

Figure 1.1: Corpus containing two trees

Then a DOP1 model could be extracted from this corpus by cutting up its trees, obtaining a *bag of subtrees* or *multiset*[2]. A *subtree* or *fragment* t is a connected subgraph of a tree τ such that: i) t consists of more than one node; ii) each non-frontier node μ in t possess either all or none of the daughters in the corresponding node in τ. Figure 1.2 shows some subtrees extracted from the corpus above.

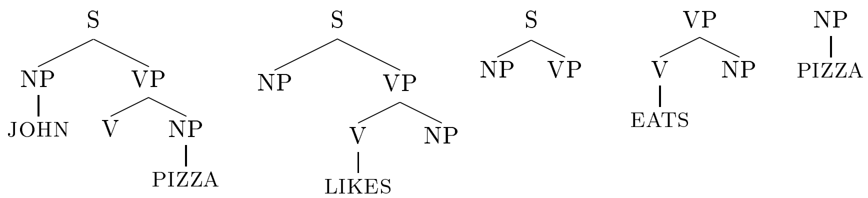

Figure 1.2: Sample of DOP subtrees extracted from the corpus

Note that not all combinations are allowed. A subtree whose root node is VP and whose only daughter is V cannot be extracted from this corpus.

Once the subtrees are extracted, they can be combined to yield the corpus trees again. Moreover, new trees can be obtained by combining subtrees in different ways. Figure 1.3 shows a derivation of JOHN EATS PIZZA.

The symbol $\circ$ is a partial function representing the *substitution* operation. Let t and u be trees such that the leftmost non-terminal of t is equal to the root node of u. Then $t \circ u$ is the tree resulting from the substitution of u in the leftmost non-terminal of t.

[2]Multisets are formally defined in Chapter 3.

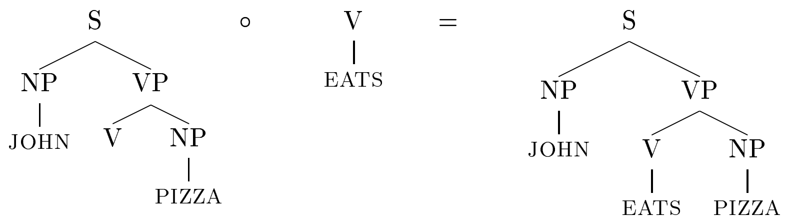

Figure 1.3: Possible derivation of JOHN EATS PIZZA

DOP1 and DOP models in general are *redundant* in the sense that they allow multiple derivations of the same tree. This is in contrast with models based on SCFGs, that permit only one derivation per tree[3]. DOP1 defines a *Stochastic Tree-Substitution Grammar* (STSG), which is a generalization of SCFGs, by taking into account also co-occurrence of context-free rules.

Given an STSG defined by a particular DOP model, a *derivation* of a tree τ with respect this STSG is a tuple of subtrees $\langle t_1, \ldots, t_n \rangle$ such that $t_1, \ldots, t_n$ are in the set of subtrees associated with the STSG and $t_1 \circ \ldots \circ t_n = \tau$. Different derivations can generate the same tree, as illustrated in Figure 1.4.

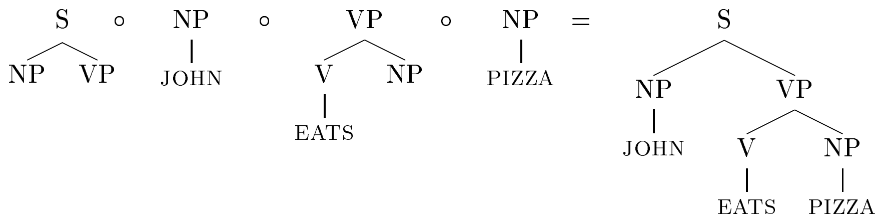

Figure 1.4: Different derivation yielding the same tree

The advantage of taking into account redundant structures as opposed to a non-redundant minimal set of rules is that more relations can be captured. The verb LIKES, for instance, takes NP and VP arguments. This relation is captured by DOP1 (see Figure 1.2), but cannot be captured by an SCFG. SCFGs and STSGs are defined in Subsection 1.1.2.

In order to fully define DOP1, we need to specify the probability estimation procedure. Let $r(t)$ represent the root node of subtree t and $f(t)$ the number of times t occurred in a bag of subtrees. Then the probability of t is estimated as:

$$P(t|r(t)) = \frac{f(t)}{\sum_{t':r(t')=r(t)} f(t')} \qquad (1.1)$$

In other words, the true probability of a subtree is estimated by taking its relative frequency in a distribution of subtrees with the same root node. Thus,

[3] To make a context-free grammar derivation unique, one usually fixes the substitutions to take place in the leftmost non-terminal leaf of a tree.

for a given corpus, several subtree distributions can be constructed. There are 8 subtrees in the VP-rooted distribution extracted from the corpus in Figure 1.1. Two of those subtrees are represented by VP $\rightarrow$ V NP and are assigned probability 2/8, the remaining subtrees are distinct and get probability 1/8.

The probability of a derivation d yielding a tree τ is then given by the probability of combining subtrees $t_1, \ldots, t_n$ such that $t_1 \circ \ldots \circ t_n = \tau$. If one assumes that the subtrees involved in the derivation of τ are independent from one another, then the probability of the joint event d is given by:

$$P(d) = \prod_{i=1}^{n} P(t_i) \tag{1.2}$$

Since there are multiple derivations for a single tree τ, the probability of τ is the probability that any of its derivations occurs. Let $\delta(\tau)$ be the set of all derivations of τ. Then, the probability of τ is given by:

$$P(\tau) = \sum_{d_i : d_i \in \delta(\tau)} P(d_i) \tag{1.3}$$

Finally, the probability of a sentence s is the probability that any of its parses occurs. A tree τ is a *parse* of string s with respect to a corpus $\mathcal{C}$ if the yield of τ equals s. The *yield* of a tree τ, written as $y(\tau)$, is the string obtained by concatenating all leaf symbols of τ. Thus, the probability of s is given by:

$$P(s) = \sum_{\tau_i : y(\tau_i) = s} P(\tau_i) \tag{1.4}$$

1.1.2 DOP grammars

In this section we define formally some concepts introduced in the previous section and discuss the stochastic power of DOP models.

A *Context-Free Grammar* (CFG) is a four-tuple $G = \langle V_N, V_T, R, S \rangle$, where V_N is a finite set of non-terminal symbols (e.g. NP), V_T is a finite set of terminal symbols (e.g. PIZZA), R is a finite set of production rules (e.g. NP $\rightarrow$ DT NN, NN $\rightarrow$ PIZZA) and $S \in V_N$ is the distinguished *start* symbol (e.g. S).

A partial-*derivation* of string $\gamma \in (V_N \cup V_T)^+$ from string[4] $\alpha \in (V_N \cup V_T)^+$ is a finite sequence of leftmost applications of rules $N \rightarrow \beta \in R$ that transforms α into γ. A partial-derivation can be seen as a concatenation of CFG-rules through a finite sequence of leftmost substitution steps.

A *derivation* is a partial-derivation that starts with the start symbol and terminates with a string consisting of only terminal symbols.

A *partial-parse* is an abstraction of a partial-derivation obtained by obscuring the rule identities of that partial-derivation. So, a partial-parse is a subtree.

A *parse* is partial-parse obtained from a derivation. Since it obscures the identities of the rule applications, it can be seen as the set of all its derivations.

[4]The symbol "+" indicates that the set $V_N \cup V_T$ has at least one element. This means that we are not dealing with empty productions, such as A $\rightarrow \epsilon$.

Let $G = \langle V_N, V_T, R, S\rangle$ be a CFG. A *Tree-Substitution Grammar* (TSG) based on G is a four-tuple $G' = \langle V'_N, V'_T, R', S\rangle$, where $V'_N \subseteq V_N$, $V'_T \subseteq V_T$, $S \in V_N$ and R' is a finite set of partial-parses of G rules over the symbols in $(V_N \cup V_T)^+$. Each element of R' is called a subtree or fragment.

CFGs and TSGs can be enriched with probabilistic information. A *Stochastic Context-Free Grammar* (SCFG) is a five-tuple $\langle V_N, V_T, R, S, P\rangle$, where V_N, V_T, R and S are defined as above and $P : R \rightarrow [0,1]$ is a probability function such that $\forall N \in V_N : \sum_{\beta: N\rightarrow\beta\in R} P(N \rightarrow \beta) = 1$.

Likewise, a *Stochastic Tree-Substution Grammar* (STSG) can be defined as a five-tuple $\langle V'_N, V'_T, R', S, P'\rangle$ that extends a TSG $\langle V'_N, V'_T, R', S\rangle$ with a probability function $P : R' \rightarrow [0,1]$, such that $\forall t \in R'$, $0 \leq P(t) \leq 1$ and $\forall N \in V_N : \sum_{t\in R': r(t)=N} P(t) = 1$.

Every DOP model extracted from a corpus defines an STSG, and every STSG can be reduced to a SCFG as follows: for each subtree t extracted from a corpus $\mathcal{C}$, define a CFG-rule $r(t) \xrightarrow{i} y(t)$ with its corresponding probability, where i is an index that uniquely identifies the internal structure of t. A grammar constructed in this way is referred to as the *SCFG underlying an STSG*.

Two stochastic grammars are called *strongly equivalent* if they generate the same strings with the same trees. STSGs and SCFGs are strongly equivalent.

Two stochastic grammars are called *superstrongly equivalent* if they are strongly equivalent and generate the same probability distribution over the trees. Bod (1993) showed that there exists a probability distribution defined by an STSG that cannot be defined by an SCFG. In other words, he showed that STSGs are stochastically more powerful than SCFGs.

Indeed, DOP grammars can be regarded as an extension of SCFGs. Let the function $h(t)$ be the *depth* of subtree t if it denotes the number of edges along the longest path from the root to a leaf of t. Then a DOP model can be reduced to an SCFG if, for all subtrees t in this model, $h(t) = 1$.

1.1.3 Computational issues

The stochastic power of DOP STSGs comes at a price. The set of subtrees that can be extracted from a corpus is so huge that the actual computation of parses become prohibitively slow. In the following we briefly describe some *implementations* of DOP models and the assumptions that had to be made in order to yield practical systems.

The task of a probabilistic parser or *disambiguator* is to find the most probable parse for a given sentence. In other words, given a sentence s, a parser *searches* for the parse τ that maximizes $P(\tau|s)$. If τ^* is such a parse, then:

$$\tau^* = argmax_\tau\ P(\tau|s) = argmax_\tau\ \frac{P(\tau, s)}{P(s)} = argmax_\tau\ P(\tau, s) \tag{1.5}$$

Let $\delta_s(\tau)$ be the set of all derivations of τ for all τ such that $y(\tau) = s$. Let $\theta^s_\tau(d)$ be the set of all subtrees t that constitute $d \in \delta_s(\tau)$, then

$$(1.6) \qquad argmax_{\tau}\ P(\tau, s) = argmax_{\tau} \sum_{d_i \in \delta_s(\tau)} \prod_{t_j \in \theta^s_{\tau(d_i)}} P(t_j)$$

The number of possible derivations for a given sentence can grow exponentially, since the number of corresponding parses can grow exponentially (Church and Patil, 1982). Thus, finding the *Most Probable Parse* (MPP) τ^* for an input sentence s can be computationally very costly. Indeed, Sima'an (1999) showed that this problem is NP-complete[5].

Bod (1993) estimated MPP by using Monte Carlo techniques. In this case, a derivation forest is created for each sentence and the MPP is estimated by sampling a large number of random derivations from this forest. The larger the sample, the more reliable the estimate. Nevertheless, Goodman (1998) showed that the number of derivations that must be sampled in order to give a reliable estimate grows exponentially with sentence length.

Sima'an (1999) devised an optimized algorithm for DOP that can compute the *Most Probable Derivation* (MPD) in time-complexity of order linear in grammar size and cubic in sentence length. The implementation of this algorithm increased parse speed by a factor of 100 for the MPD calculation.

The MPD is obtained by assuming that most of the probability mass of a given parse is concentrated on a single derivation. Finding such a derivation would therefore result in finding most of the time the most probable parse. Formally, the MPD approximation can be stated as:

$$(1.7) \quad \tau^* = argmax_{\tau} \sum_{d_i \in \delta_s(\tau)} \prod_{t_j \in \theta^s_{\tau(d_i)}} P(t_j) \ \approx\ argmax_{d_i} \prod_{t_j \in \theta^s_{\tau(d_i)}} P(t_j)$$

Sima'an's parser implements a DOP grammar as a rule-based STSG. Most of the results reported in Chapter 4 were obtained through the MPD approximation and all of them were carried out with Sima'an's parser.

1.1.4 Performance on benchmark corpora

DOP1 model has delivered promising results when tested in benchmark corpora. For brevity, we will cite only recent results obtained with the Wall Street Journal (WSJ) corpus (Marcus et al., 1995), a treebank with over 40,000 sentences, some of them with over 100 words (average sentence length: 23 words).

Bod (2001) reports results obtained with DOP1[6] in the WSJ corpus. After training with section 2 through 21 (approx. 40,000 sentences) the model was tested with section 23 (2,416 sentences, each having less than 100 words).

[5] Finding a deterministic polynomial-time algorithm for this problem is very unlikely.

[6] The extraction procedure for these experiments were slightly modified. For each depth 1–14, 400,000 trees were randomly sampled from the bag of possible subtrees. We describe issues in the extraction procedure in section 1.1.5.

DOP1 achieved 89.7% in Labeled Precision and 89.7% in Labeled Recall[7]. These results slightly outperform state-of-the-art parsers, such as the one described by Charniak (1999). As mentioned earlier, alternative DOP models, such as the Tree-gram model, have improved on DOP1 results on other domains (Sima'an, 2001b).

1.1.5 Shortcomings

Despite good performance, DOP1 suffers from some major drawbacks, leaving room for improvements. We will describe shortly three of them.

Weak lexicalization

DOP subtrees miss some relations between words that other models are able to capture. Let $l(t)$ represent a set containing the leaf nodes of subtree t. A subtree t is *lexicalized* with respect to an STSG $G' = \langle V'_N, V'_T, R', S, P' \rangle$ if $\exists w \in l(t) : w \in V'_T$. One feature of DOP subtrees is that they are *frontier-lexicalized*, meaning that the lexical items, if any, are present only at leaf nodes.

This is in contrast to *head-lexicalized* models (Collins, 1997). A lexical item w is called the *head* of constituent τ if w is the "most important word" in τ (e.g. the main verb in a Verb-Phrase, the main noun in a Noun-Phrase)[8]. In these models, information about lexical items is attached to non-terminal nodes as well as with terminal nodes. As a consequence, head-lexicalized models can capture relations between words even if their rules or subtrees are not lexicalized. DOP subtrees, on the other hand, lose lexical information as soon as they become unlexicalized during the extraction procedure.

Consider the sentence HE GAVE THE BOY LESSONS. Two possible analysis for this sentence are given in Figure 1.5.

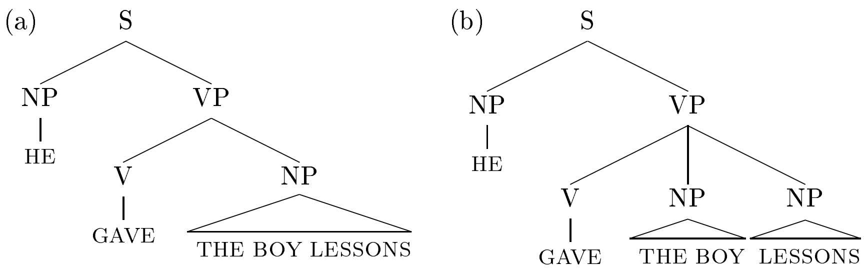

Figure 1.5: Two possible analysis of HE GAVE THE BOY LESSONS

To disambiguate this sentence and obtain the preferred parse 1.5(b), the parser must know the relative probabilities of the competing VP-rules, namely

[7]The concepts of recall, precision and other evaluation metrics are described in Chapter 4.

[8]The concept of head is a bit loose and can vary across linguistic schools. In the probabilistic parsing tradition, the head words are commonly determined by a head percolation table defined by Magerman (1995).

VP $\rightarrow$ NP and VP $\rightarrow$ NP NP. These probabilities, however, can vary from verb to verb. Head-lexicalized rules are able to capture these differences.

Rule r	$P(r)$	*Rule r*	$P(r)$	*Rule r*	$P(r)$
VP $\rightarrow$ V NP	0.6	VP$_{sell}$ $\rightarrow$ V NP	0.7	VP$_{give}$ $\rightarrow$ V NP	0.3
VP $\rightarrow$ V NP NP	0.4	VP$_{sell}$ $\rightarrow$ V NP NP	0.3	VP$_{give}$ $\rightarrow$ V NP NP	0.7

Table 1.1: Example of head-lexicalized and -unlexicalized VP-rules.

Table 1.1 shows an artificial set of VP rules. In the first column, there are unlexicalized rules. They represent only an average over the *subcategorization* preferences of the verbs. Columns 2 and 3 show that, depending on the verb, the preferences can even be reversed: the verb SELL takes most of the times only one object whereas GIVE takes two.

A head-lexicalized model would be able to capture these preferences and assign the correct structure to the sentence, since it would choose the VP-rule according to a distribution conditioned on GIVE. DOP, on the other hand, could only do so by change. If the corresponding STSG contains subtrees anchored with GIVE, then it might capture some of the lexical dependencies. But when these trees are not present, the derivation will be blind, since unlexicalized nodes higher up in the subtrees do not have access to subcategorization information that is specific to the verb under consideration.

Tree-grams (Sima'an, 2001b) improved on DOP1, among other reasons, because it extracts subtrees from an corpus having utterance-analyses enriched with head information, generating thereby a head-lexicalized STSG.

Restricted robustness

DOP1 extraction procedure generates subtrees that are equivalent to the context-free rules underlying an annotated treebank[9] and combinations thereof. If VP $\xrightarrow{i}$ NP NP and VP $\xrightarrow{i}$ NP PP are the only VP-rules in a given treebank, then neither rules of the form VP $\xrightarrow{j}$ NP nor rules of the form VP $\xrightarrow{j}$ NP NP PP would be assigned probabilities in the corresponding STSG, since their frequency is zero. This greatly restricts the number of rules that can be generated and increases the chance of failing parses.

Markov grammars (Charniak, 1999) present a robust and efficient solution to this problem. Rather than generating and storing all subtrees as rules of an STSG grammar, Markov grammars allow building rules on the fly through the combination of rules and parts of rules. For lexicalized DOP models, Markov grammars are even more important, since the number of head-lexicalized rules would quickly explode.

Given a rule V$_{sell}$ $\rightarrow$ V NP, a Stochastic Lexicalized Context-Free Grammar (SLCFG) using DOP1 estimator would assign to it a probability P(V NP|VP,sell): the number of times the right-hand side (rhs) V NP was seen in the distribution

[9] Charniak coined the term "treebank grammars" to refer to CFGs extracted from corpora.

conditioned on the left-hand side (*lhs*) VP SELL. If the joint event describing the *rhs* was not seen, the estimate is zero.

In contrast, a Markov grammar would *generate* a probability estimate given by $P(\text{V}|\text{VP,sell}) \times P(\text{NP}|\text{VP,V,sell})$. The generation process goes as follows: first generate the probability of the head-child H (child that contains the head), given its parent node P and the head-word h. In the previous example, $H = \text{V}$, $P = \text{VP}$ and $h = \text{SELL}$. Next, generate the probabilities of left (l_j) and right (r_j) modifiers of the head-child. In the example, $r_j = \text{NP}$. The final probability is obtained by making a Markov assumption[10]. The lexical sensitivity of Markov models comes from conditioning probabilities on head-lexicalized non-terminals.

Rule r	SLCFG $P(rhs\|lhs, h)$	Markov Grammar $\prod_i P(X_i\|[H,]P, h[, l_j, r_j])$
$\text{VP}_{sell} \rightarrow$ V NP NP	P(V,NP,NP|VP,sell)	$\ldots \times P$(NP|V,VP,sell,NP)
$\text{VP}_{sell} \rightarrow$ V NP PP	P(V,NP,PP|VP,sell)	$\ldots \times P$(PP|V,VP,sell,NP)
$\text{VP}_{sell} \rightarrow$ V NP NP PP	0	$\ldots \times P$(PP|V,VP,sell,NP,NP)

Table 1.2: Probabilities from a Stochastic Lexicalized CFG (SLCFG) and Markov grammar. When a rule is not found, SLCFG assigns probability zero, whereas a Markov grammar builds a new probability estimate from rule pieces.

Table 1.2 compares the probability estimations of VP-rules using a SLCFG and a Markov grammar. Column 1 shows SLCFG-rules and columns 2 and 3, the probabilities estimates for each grammar. Suppose the first two rules in column 1 were the only VP-rules occurring in a given treebank. Then, the rule in row 3 would be assigned probability zero according to the SLCFG. The Markov grammar, on the other hand, would assign a non-zero probability, since it could obtain an estimate of the rule by looking at the rule in row 2 and making a 1st-order Markov assumption[11].

Tree-grams (Sima'an, 2001b) used Markov grammars to increase the robustness of DOP models, achieving better performance than DOP1.

Estimator: biased and inconsistent

The estimator used by DOP1's has been shown to be biased and inconsistent in a sense that will be defined below. First, we will give an intuitive example (Bonnema et al., 1999) showing why the estimator should be biased. In the following, we present a result, due to Johnson (2002), proving this property.

Assume a treebank with only three trees as illustrated in Figure 1.6(a–c). The preferred analysis to string *ab*, given the treebank, should be the one in Figure 1.6(b). However, this is not the analysis chosen by DOP1. There are 134 different subtrees of category A. Thus, each will be assigned probability 1/134. Category X, however, has only 4 different subtrees, each with probability 1/4.

[10] Markov processes are explained in Chapter 2.

[11] In other words, $P(\text{PP}|\text{V,VP,sell,NP,NP}) \approx P(\text{PP}|\text{V,VP,sell,NP})$.

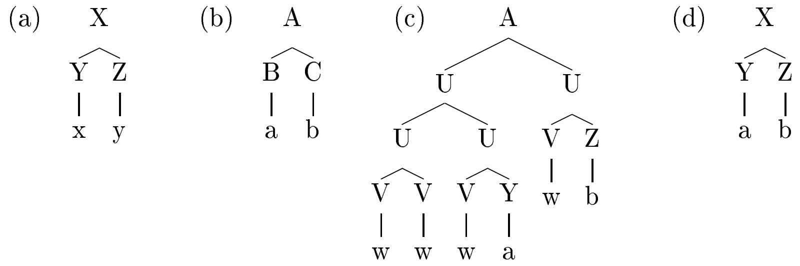

Figure 1.6: Example treebank (a–c) showing DOP bias towards larger trees. For input string ab, DOP1 chooses tree (d) over tree (b) as the preferred analysis.

Tree 1.6(d) is preferred over tree 1.6(b), since it is more likely according to DOP1 estimation procedure. Even the sum of the probabilities of all four derivations of 1.6(b) is lower than the probability of a single derivation of 1.6(d).

The reason for this distortion lies in the exponential nature of the tree extraction procedure. For a constituent τ whose root node μ has k daughters, the size of the set of μ-rooted subtrees is given by the recursive equation

$$|\sigma(\tau)| = \prod_{i=1}^{k}(|\sigma(\tau_i) + 1)| \tag{1.8}$$

As a consequence, bigger trees of a given category contribute in a disproportionate manner to the probability mass of that category, shrinking the relative frequency of smaller trees.

Johnson (2002) shows, with a counterexample, that the DOP1 estimator is biased and inconsistent. Let Θ be the space of all possible subtree probability assignments. An estimator ϕ is a function from a vector X of n subtrees to a parameter value $\phi(X) \in \Theta$. For a corpus $\mathcal{C}$ from which n subtrees can be extracted, the DOP1 estimator procedure can be represented as $\phi(X) = (P(t_1), \ldots, P(t_n))$, with P defined as in Equation 1.1.

Intuitively, an estimator ϕ of a true parameter θ^* is biased if its *expected value* $E_{\theta^*}(\phi(X))$ is different from θ^*. It is inconsistent if, as the sample size grows, the corresponding sequence of probability distributions estimated by ϕ does not converge to the distribution θ^* that generated the data.

Let $\theta^* \in \Theta$ be the true probability assignment of subtrees that generated the trees in corpus $\mathcal{C}$. If $t_1, \ldots, t_n$ represent n fragments extracted from $\mathcal{C}$, then $\theta^* = (P^*(t_1), \ldots, P^*(t_n))$, where P^* assigns the "true" probabilities to the fragments. Let P_{θ^*} be a probability distribution on parse trees. For tree τ, $P_{\theta^*}(\tau)$ is the probability of τ given by θ^*. Similarly, let $P_\phi(\tau)$ be the probability estimate of τ given that the subtree probabilities are estimated by ϕ.

A *loss function* of ϕ given θ^* is defined as $\mathcal{L}(\theta^*, \phi(X)) = \sum_\tau P_{\theta^*}(\tau)(P_\phi(\tau) - P_{\theta^*}(\tau))^2$. It measures the loss caused by the error in the estimate ϕ of θ^*. The *risk* of an estimator ϕ at θ^* is its expected loss $E_{\theta^*}(\mathcal{L}(\theta^*, \phi(X)))$. Now we can define properly bias and inconsistency:

- An estimator ϕ is *biased* if $P_{E_{\theta^*}(\phi)} \neq P_{\theta^*}$;
- An estimator ϕ is *inconsistent* if $lim_{n\to\infty} E_{\theta^*}(\mathcal{L}(\theta^*, \phi(X))) \nrightarrow 0$.

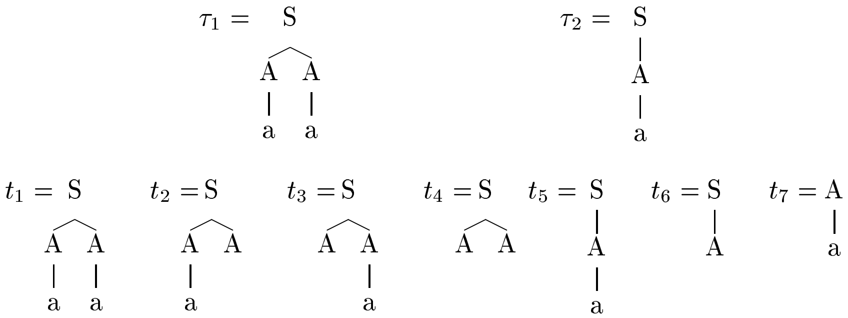

Figure 1.7: Trees τ_1, τ_2 and corresponding subtrees $t_1, \ldots, t_7$ in DOP1 model

Assume the simple treebank with trees τ_1 and τ_2 and corresponding fragments as shown in Figure 1.7. Suppose the true probabilities of subtrees $t_1, \ldots, t_7$ in Figure 1.7 are 0 except for:

$$\begin{aligned} P^*(t_4) &= p, \\ P^*(t_6) &= 1-p, \\ P^*(t_7) &= 1. \end{aligned}$$

Then, the true probabilities of τ_1, τ_2 are given by $P_{\theta^*}(\tau_1) = p$, $P_{\theta^*}(\tau_2) = 1-p$. The expected number of occurrences of tree τ_1 in a sample of size n is np and the expected number of occurrences of tree τ_2 is $n(1-p)$. Thus, the expected frequency of subtrees in a sample of size n is

$$\begin{aligned} E(f(t_i)) &= np && \text{for } i = 1,\ldots,4; \\ E(f(t_i)) &= n(1-p) && \text{for } i = 5,6; \\ E(f(t_7)) &= n+np \end{aligned}$$

Thus, the expected estimated probabilities of subtrees according to DOP1 estimator is given by

$$\begin{aligned} E(P(t_i)) &= \frac{p}{2+2p} && \text{for } i = 1,\ldots,4; \\ E(P(t_i)) &= \frac{1-p}{2+2p} && \text{for } i = 5,6; \\ E(P(t_7)) &= 1 \end{aligned}$$

The estimated probabilities of trees τ_1 and τ_2 using the expected subtree probabilities is

$$P_{E_{\theta^*}(\phi)}(\tau_1) = \frac{2p}{1+p}$$

$$P_{E_{\theta^*}(\phi)}(\tau_2) \quad = \quad \frac{1-p}{1+p}$$

That DOP1 estimator is biased follows from the fact that $P_{E_{\theta^*}(\phi)}(\tau_1) \neq P_{\theta^*}(\tau_1)$ for $0 < p < 1$. Inconsistency follows from the observation that, as the sample size grows, $E_{\theta^*}(\mathcal{L}(\theta^*, \phi(X)))$ does not approach zero. Thus, the probability estimates for subtrees in DOP1 do not converge to the true probability values that generated the data.

A biased estimator is not necessarily bad. Efficient estimators may be obtained by trading variance for bias (Wonnacott and Wonnacott, 1990). Inconsistency however is considered a fatal flaw. Fortunately, a new estimation procedure for DOP1 and DOP models in general has been put forward (Bonnema et al., 1999; Bonnema and Scha, 2002). In the following, we describe how the new estimator was derived and highlight some of its properties.

1.2 Bonnema's estimator

The bias and inconsistency of DOP1 described in the previous section suggest that the model would yield disastrous results. This is not the case. The reason for that is that DOP models have been rarely implemented in their full length, since the number of subtrees is too large. Several heuristics have been suggested to constrain the number of subtrees (Sima'an, 1999). Promising results have been obtained by varying parameters such as maximal number of words, maximal subtree depth or maximal number of substitution nodes. By restricting the subtree space, previous DOP models have counterbalanced the estimator's bias, since these restrictions favor smaller subtrees over larger ones.

Optimal parameter values, however, can only be found by a corpus-dependent time-consuming process of varying parameters and comparing results. For every new corpus, the same process has to be repeated. Moreover, this is an *ad-hoc* strategy that goes against DOP's original intuition of taking all fragments and letting the statistics decide which subtrees are the relevant ones.

The main motivation to look for an alternative estimation procedure for DOP is to ground the theory in more firm soil, increase predictive power and, as a consequence, obtain better results.

1.2.1 Maximum-Likelihood: not the answer

One might be tempted to say that DOP1 estimator is biased because it is not a Maximum-Likelihood[12] (ML) estimator, since it does not maximize the likelihood of the data. Suppose we are given a treebank $\mathcal{C} = \tau_1, \tau_2$. Then the probability assignment that maximize the likelihood of $\mathcal{C}$ is given by $P(\tau_1|root(\tau_1)) = P(\tau_2|root(\tau_2)) = 1/2$ and zero to all other subtrees. This is not satisfactory in the context of parsing natural language, since such probability assignment does not allow for generalization. DOP models distribute the probability that

[12]ML estimators are consistent and unbiased. They are described in detail in Chapter 2.

would be given to the corpus trees among all its subtrees, enabling the model to produce parses not seen previously in the treebank. An ML estimator would not allow for that and, therefore, should be discarded.

1.2.2 Gathering subtree evidence

In order to estimate the probability of a subtree from a given corpus, we have to gather evidence of the frequency of that subtree in the corpus and count how often that subtree is used in derivations.

We assume that the corpus has been generated by an STSG and that every tree in the corpus is the result of derivations using STSG elementary trees. Since corpora in general are annotated only with phrase-structure information, we do not know which derivations participated in the construction of a tree.

Bonnema (1999) takes this lack of information seriously and assumes that all possible derivations play a role and that they are uniformly distributed. In other words, each corpus tree is regarded as the set of all possible derivations and each derivation is equally likely. Evidence for a specific fragment then comes from two sources: the relative frequency of the tree (*constituent distribution*) and the fraction of the derivations of this tree that contains a substitution of the fragment (*fragment distribution*).

1.2.3 Estimation procedure

We will follow here Bonnema (1999) description. A fragment t is said to *start* a parse tree τ if there exists a derivation $t_1 \circ \ldots \circ t_n$ of τ such that $t = t_1$. If a fragment starts a derivations it is called an *initial fragment*. Given a subtree t and a constituent τ, let $\phi(t, \tau)$ be the fraction of all derivations of τ that *start with* t. Let $\delta(\tau)$ be the set of all possible derivations of a constituent τ. Then,

$$\phi(t,\tau) = \frac{|\{d_j \in \delta(\tau) : t_{1j} \circ \ldots \circ t_{d_j j} = \tau\}|}{|\delta(\tau)|} \qquad (1.9)$$

with $t_{1j} = t$.

For any derivation of τ, a non-root node is either internal to a subtree or a substitution node. Any combination is allowed. So, the cardinality of $\delta(\tau)$ equals the powerset of all non-root nodes in τ. Let $N(\tau)$ be the number of non-root non-terminals of τ. Then, $|\delta(\tau)| = 2^{N(\tau)}$. If a fragment t has $N(t)$ non-root nodes, then only $N(\tau) - N(t)$ nodes remain available for substitution. Since each of these nodes can also be either internal or a substitution node in a derivation, there are $2^{N(\tau)-N(t)}$ derivations involving node t. Substituting in equation 1.9 gives

$$\phi(t,\tau) = \frac{2^{N(\tau)-N(t)}}{2^{N(\tau)}} = \frac{2^{N(\tau)}2^{-N(t)}}{2^{N(\tau)}} = 2^{-N(t)} \qquad (1.10)$$

Interestingly, the probability that a fragment starts a derivation depends only on the fragment's complexity. This a direct consequence of the assumption of uniform distribution among derivations. In the absence of more precise information about the shape of the derivation distribution, the fragment itself decides how much probability it should get. So, another assumption underlying this new estimator is that the probability of a fragment is invariant across derivations of different parse trees, since the estimate depends only on the fragment's complexity and not on the surrounding fragments with which it interacts.

The second factor that defines the subtree probability is given by the probability that the subtree t of category A occurs in the corpus. This is given by the relative frequency estimator

$$F(t) = \frac{f(t)}{\sum_{t':r(t')=r(t)} f(t')} = \frac{f(t)}{f(A)} \tag{1.11}$$

which is the DOP1 estimator. Combining the two information sources, namely, fragment and constituent distributions, yields a new DOP estimator given by

$$P(t) = 2^{-N(t)} F(t) \tag{1.12}$$

This estimate can be assigned to elementary trees in an STSG extracted from a corpus. Then, the probability of a parse tree with n possible derivations $d_1, \ldots, d_n$, each with m_j subtrees $t_{1j}, \ldots, t_{m_j j}$ becomes:

$$P(\tau) = \sum_{j=1}^{n} \prod_{i=1}^{m_j} 2^{-N(t_{ij})} F(t_{ij}) \tag{1.13}$$

The new estimator proposed by Bonnema yields a new DOP model. For ease of reference, we will henceforth call this new model **Bon-DOP**.

1.2.4 Promising properties, bad results

The Bon-DOP estimator presents interesting properties. It can be viewed as a DOP1 estimator with a correction factor, $2^{-N(t)}$, for subtree size. Large subtrees have their estimate downgraded and short subtrees are upgraded[13].

A quick review on the examples described in Section 1.1 shows that the new estimator improves on DOP1. As to the treebank in Figure 1.6, Bon-DOP assigns the correct parse to input string ab. Moreover, it gives correct probabilities to the fragments in Figure 1.7.

There is no general proof so far that the Bon-DOP estimator is unbiased and consistent. Nonetheless, it is so for treebanks that are assumed to be generated by SCFGs (Bonnema and Scha, 2002). Given this assumption, Bon-DOP estimator is consistent because its probability estimates can be reduced to the true probabilities of the SCFG that generated the data.

Bon-DOP is a member of a class of DOP models represented as a weighted sum of products of the relative frequencies of the subtrees participating in each derivation. An SCFG can be seen as the particular case where the weight is 1 to derivations corresponding to CFG derivations and 0 to all other derivations.

[13]The effect of taking more probability from large trees than from short trees is the same as enlarging the relative frequency of short trees in the distribution, increasing their estimates.

Despite the promising properties of the new DOP estimator, empirical results have been disappointing (Sima'an, personal communication). We believe the reason for that lies in the simplifying assumptions behind the new estimator.

1.2.5 Simplifying assumptions: psycholinguistic evidence

The main assumption behind Bon-DOP is that all derivations of a given parse tree are equally likely. We argue that this is not a good assumption given current psycholinguistic results about the way humans parse natural language. Looking at psycholinguistic evidence is justifiable to the extent that humans are the fastest and most robust natural language parsing systems known thus far.

It is important to notice that different derivations in Bon-DOP model will end up having different probabilities after summing up evidence from the whole treebank. This happens because the second factor in Equation 1.12, $F(t)$, is a relative frequency estimator that takes into account frequency information from the whole subtree distribution.

The problem here is in the derivation of the first factor, $2^{-N(t)}$. For a fixed tree τ and an initial fragment t, the evidence for t is counted as the number of times t could have started a derivation of τ. If the "true" STSG that generated the treebank have some tendency to give more probability mass to shorter derivations having longer subtrees – because this would capture more linguistic structure, for example – Bon-DOP estimator would miss the target.

Psycholinguistic studies have been showing that there are constraints as to the types of derivations humans make[14]. People tend to parse sentences *incrementally* (Crocker, 1996). An incremental parser process every input word sequentially, from *left-to-right*, in a *word-by-word* basis, inserting every new lexical item into a *permanently connected* structure[15].

Incremental parsing contrasts with *head-driven* parsing, where the attachment of a phrase to the current structure is delayed until its head is reached (Abney, 1987). While the head is not reached, unconnected phrases have to be stored, contradicting the incremental property of connectedness. Incrementality would bring two advantages to a cognitive parsing system: i) it allows partial interpretation; ii) it reduces memory load, since one does not need to keep large sequences of unstructured items in working memory.

Support for incrementality comes from studies showing that people are able to recall the meaning of partial parses as soon as new words are read in (Marslen-Wilson, 1973). Garden-path[16] effects also depose in favor of this view.

In order to maintain a fully connected structure throughout parsing, some input words will have to anchor subtrees whose heads have not yet been read in the input (*headless projections*). Unlexicalized subtrees with headless projection

[14] We do not claim that 'derivations' or 'trees' are the representations of parsing structures in the cognitive system. Our aim is simply to illustrate that, at least from a psychological point of view, assuming uniform distribution for derivations is not a good way to go.

[15] 'Structure' can be an edge in a chart (see also Brants and Crocker (2000)).

[16] Sentences that mislead the human parser into failed analyses: "The man who cooks ducks out on weekends."

nodes are called *connection paths*. Figure 1.8 illustrates a connection path in a partially parsed structure when the processing of input sentence HE THINKS STEEPER PRICES HAVE COME ABOUT BECAUSE ... is halted at word STEEPER. Note that, in order to insert STEEPER into the current representation, the parser needs to hypothesize two headless projections, namely, the NP which STEEPER modifies and the S projection.

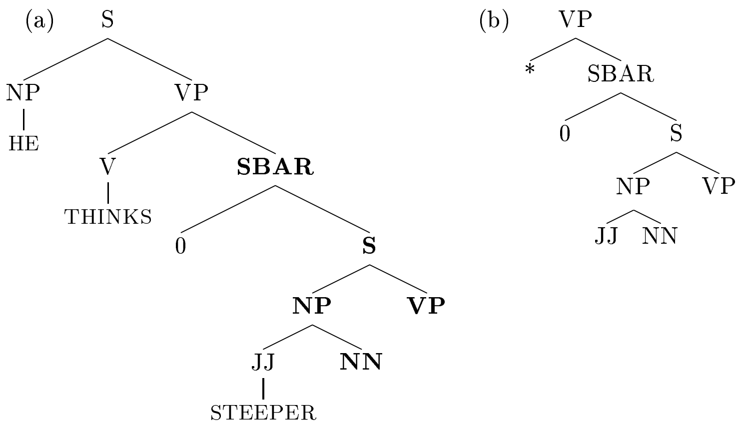

Figure 1.8: Partial tree representation (a) during parsing with an incremental disambiguator. The current input word is STEEPER. In order to maintain connectedness, the subtree which STEEPER anchors needs to hypothesize headless projections (NP and S nodes). Subtree (b) is a connection path representing the hypothesized projections.

Lombardo and Sturt (2002) assessed in the Wall Street Journal corpus some of the consequences of using an incremental parsing strategy. Instead of building an incremental parser, they devised a *parsing simulation algorithm* that takes a parse tree and identifies the word-by-word steps taken by an incremental parser in order to build the tree. The results showed that 80% of the connection paths do not involve any headless projection. So, in the processing of realistic English, 80% of the words could be attached without involving headless hypotheses. Moreover, only 0.13% of the paths involved more than 2 headless projections.

These results suggest that derivations not yielding incremental parses are less likely than derivations yielding incremental ones: derivations involving subtrees with more than 2 headless projections would be less likely than derivations involving subtrees with less than 2 headless projections. This contradicts the view that derivation distributions are uniform.

1.2.6 Limits of the uniform distribution

The direct result of the uniform distribution assumption is a factor $2^{-N(t)}$ that gets multiplied by the probability estimate of DOP1. Thus, according to Bon-DOP estimation procedure, large subtrees will see their probability estimates shrink by the simple fact that they are large. Conversely, small subtrees will have their estimates enlarged by the simple fact that they are small.

For subtree distributions in general, the frequency of large subtrees is smaller than the frequency of shorter ones. This is so because it is much more likely to see a joint event containing few elements than a joint event containing many elements. It is much more likely to see subtrees whose leaf nodes are THE MAN than subtrees whose yield is THE MAN WITH GLASSES THAT I TOLD YOU.

Thus, there are two additive and opposite effects working on Bon-DOP estimation: tree size and relative frequency. As a result, large subtree probabilities are underestimated and small subtree probabilities are overestimated.

To ground the discussion, consider the corpus in Figure 1.9(a–c) with 17 constituents. The correct analysis for string *abcdefgh* should intuitively be the one in Figure 1.9(a). Instead, Bon-DOP chooses the parse in Figure 1.9(d)[17].

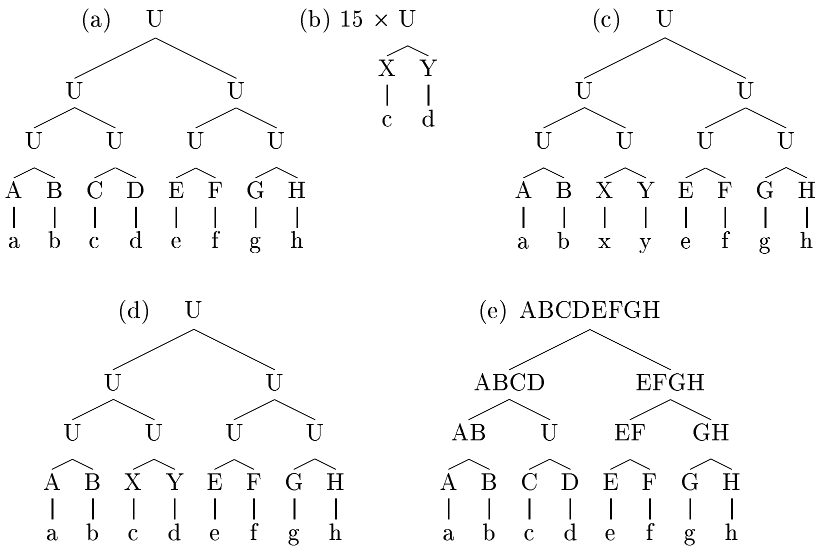

Figure 1.9: Example treebank(a–c) illustrating Bon-DOP bias towards small trees. Correct parse to string *abcdefgh* should be given by (a). The model choses (d) instead. The same behaviour is observed with less recursive trees (e).

Bon-DOP gives an excessive amount of probability mass to small trees. As a result, a derivation that was actually seen in the corpus (the parse tree yielding *abcdefgh*) becomes less likely than a new derivation involving the fragment in Figure 1.9(b). In contrast, DOP1 estimator finds the correct analysis.

One might argue that the trees in Figure 1.9(a–c) are exceedingly recursive and would never occur in an annotated corpus anyway. Furthermore, recursion

[17] We do not compare derivation probabilities here since there are 2^{14} possible derivations for the parse yielding *abcdefgh*. The results were obtained with Sima'an's (1999) disambiguator and refer to both MPD and MPP (Monte Carlo sample size: 25,000).

greatly enlarges the distribution on U, resulting in even less probability mass to big trees. To address this issue, consider a corpus analogous to the one in Figure 1.9(a–c) but containing trees with non-terminals labelled as the ones in Figure 1.9(e). This is a less recursive, more linguistically motivated corpus. The results, however, remain unchanged: Bon-DOP errs, DOP1 guesses it right.

More evidence for Bon-DOP's bias towards smaller trees comes from Bonnema et al. (2002). Suppose a corpus $\mathcal{C}$ consisting of n balanced binary trees of category S. Suppose further that one tree, τ_1, has depth 6 and the remaining $n-1$ trees $\tau_2, \ldots, \tau_n$ have depth 5. Let $n = 1,000$. Bonnema shows that, when using DOP1 estimator, 99.8% of the probability mass goes to subtrees of depth 6; and when using Bon-DOP estimator, more than 99.9% goes to subtrees of depth *less* than 6. In other words, trees of depth 6 get less than 0.1%, according to Bon-DOP estimator. In an unbiased distribution, one would intuitively expect that approximately 0.1% of the probability would go to trees of depth 6.

Thus, it appears that the factor $2^{-N(t)}$ in Bon-DOP causes an inversion in the bias of the probability estimates when compared to DOP1. In the next section, we motivate a new estimation method that, we believe, takes a more moderate view on how the probabilities should be distributed among subtrees.

1.3 Back-off as a theoretical alternative

We argued that Bon-DOP estimator presents some undesired properties and that this is a direct consequence of the assumptions used to derive it. We presented psycholinguistic evidence supporting the view that Bon-DOP assumptions are not reasonable. Nevertheless, the reason why those assumptions were made in the first place is that we lack information about the distribution of derivations. A corpus is not annotated having an STSG in mind. There are two points, however, that we have to keep in mind: i) DOP1 is biased towards large trees; ii) Bon-DOP assumption of uniform distribution is not reasonable. We believe it is possible to improve the two previous models by tackling i) and ii) in the following way: let us change the way derivations are treated probabilistically.

1.3.1 Back-Off DOP

Consider the derivations illustrated in Figure 1.10. The asterisk symbol illustrates a substitution site, so that every derivation can be uniquely represented by a tree decorated with '*'. According to the previous models, these derivations are independent events, since the probability of each one is calculated as the product of the probabilities of the fragments they are made of.

Following a suggestion by Sima'an (personal communication), we propose to consider these derivations as "back-offs"[18] of each other. In other words, we view

[18]Back-off is a smoothing technique commonly used in language modelling research. Its main goal is to smooth string distributions by reserving probability mass for previously unseen events, adding robustness to the system. Here we use back-off as a means of transferring probability mass across derivations. Chapter 2 describes several smoothing techniques.

longer derivations (with smaller subtrees) as back-up derivations that are only activated if a shorter derivation (with larger subtrees) is not available. In this sense, derivations 1.10(b-c) are back-offs of derivation 1.10(a), and derivation 1.10(d) is a back-off of derivations 1.10(b-c).

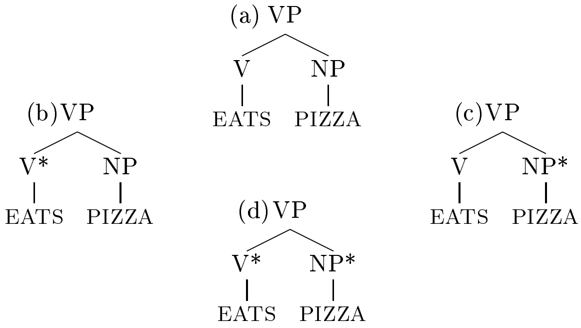

Figure 1.10: Back-off derivations. '*' indicates substitution sites.

The intuition is that subtree events extracted from a treebank are not disjoint, but related to each other, since the occurrence of a large tree t will imply the occurrence of smaller subtrees t' which are related to t. Note that the relation does not work the other way around: the occurrence of a small tree does not imply the occurrence of a larger subtree in which the small one is contained. We hypothesize that this dependence relation can be modelled by means of back-off.

Shorter derivations are able to capture more linguistically relevant dependencies than longer ones because they are made up of larger subtrees. Thus, they should be given precedence. This imposes an ordering in the derivation space. To create this ordering, we need to transfer probability mass across derivations: shorter derivations must reserve some probability mass to their back-offs and this is done by means of discounting. The transfer happens at the level of subtree distributions. Large subtrees have to give up some probability in order to allow shorter subtrees to work as their back-offs. The process follows iteratively until the shortest subtrees are reached.

In Figure 1.10, the VP-rooted subtree represented in 1.10(a) transfers some mass to the VP-rooted subtrees in 1.10(b) and 1.10(c). These, in turn, give some mass to the unlexicalized VP-rooted tree in 1.10(d).

Transferring probability in this way presents two important properties: i) probability mass from larger subtrees is transferred to smaller subtrees; ii) derivations are no longer equally likely. Because this new estimation procedure tackles some of the shortcomings of previous models, we expect to obtain better probability estimates for fragment and derivation distributions. As a consequence, we expect to increase parse accuracy.

The new estimation procedure yields a new DOP model. We will henceforth refer to this model as Back-Off DOP or **BO-DOP**.

The following chapters are organized as follows: **Chapter 2** discusses some techniques used in language modelling which partially inspired the new approach proposed in this thesis; **Chapter 3** introduces the Back-Off DOP model and details a simplified version we implemented; **Chapter 4** describes implementation details and empirical results; **Chapter 5** conclude this work and discuss further directions.

Chapter 2

Smoothing techniques in language modelling

Smoothing has been widely used in Natural Language Processing (NLP). It is an important element of probabilistic language models, enabling NLP systems to handle the omnipresent problem of data sparseness. No matter how large the training set is, there are always words or sequences of words that were never seen before. NLP systems should be able to tackle these situations and smoothing techniques provide those systems with just that.

Language modelling can be regarded as a spin-off of speech recognition research. In the following, we introduce the concept of language models in the context of speech recognition. Next, we illustrate why the maximum-likelihood estimator is not appropriate for language modelling and introduce some alternative estimators. Finally, we describe how these new procedures can be combined to yield robust estimators, such as the back-off estimator.

2.1 Language models

2.1.1 Speech recognition

In the probabilistic tradition of speech recognition research, one aims at identifying the most probable sentence that can account from a sequence of acoustic signals identified by an acoustic processor (Jelinek, 1999).

Let $A = a_1, \ldots, a_m$ be the acoustic evidence captured by an acoustic processor, where a_i is a symbol taken from some alphabet. It is on the basis of acoustic evidence A that the recognizer will make its decision about which words were spoken. Let $W = w_1, \ldots, w_l$ represent a sequence of words belonging to a fixed and known vocabulary. If $P(W|A)$ is the probability that the words W were spoken given evidence A, then the task of the recognizer is to find the most likely sequence of words $\hat{W}$ given the acoustic evidence A. This can be restated by Bayes' theorem as:

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)} \tag{2.1}$$

Thus, one searches for a sequence of words $\hat{W}$ such that

$$\hat{W} = argmax_W P(W|A) \tag{2.2}$$

Substituting Equation 2.1 in 2.2 and noting that the maximization process does not depend on $P(A)$, we have:

$$\hat{W} = argmax_W P(A|W)P(W) \tag{2.3}$$

Thus the speech recognition problem can be split into two stochastically independent subproblems: estimating $P(A|W)$ (the *acoustic model*) and estimating $P(W)$ (the *language model*). A language model is formulated as a probability distribution over sequences of words. It should assign probabilities such that grammatical and common sentences are more likely than awkward and uncommon ones. In a reasonable language model, P(JOE EATS PIZZA) $>$ P(PIZZA EATS JOE) $>$ P(EATS JOE PIZZA).

2.1.2 Markov models

The probability of a sequence[1] of l words can be expressed as

$$P(W) = P(w_1, \ldots, w_l) = \prod_{i=1}^{l} P(w_i|w_1, \ldots, w_{i-1}) \tag{2.4}$$

As the size of the conditioning history $w_1, \ldots, w_l$ increases, it gets harder and harder to obtain reliable estimates. One common procedure is to restrict the history to a fixed size. The product of these estimates is then taken as a first approximation for the probability of the sentence. When the history is restricted to the previous word, Equation 2.4 becomes

$$P(W) \approx \prod_{i=1}^{l} P(w_i|w_{i-1}) \tag{2.5}$$

Sentences can be regarded as sequences of discrete objects (e.g. words) emitted with a given probability at a given time by some output producing process (e.g. grammar). Processes with those characteristics can be modelled by *Markov models*. A Markov model is defined by a set of probability transitions and two assumptions: limited history and time invariance. A Markov model that takes into account the last n words is called a nth-order Markov model or n-gram. For n-grams in general, an event is a sequence of n words $w_{i-n+1}, \ldots, w_i = w_{i-n+1}^{i}$ appearing together in the training data. If $n = 2$, the model is called a *bigram* model; if $n = 1$, *unigram* model. Equation 2.5 specifies a bigram model.

[1] The probability of a joint event can be rewritten as a product of conditional probabilities. This rewriting is referred to as the *chain rule*. For a *proper* language model, $\sum_W P(W) = 1$.

2.2 Maximum-Likelihood

The next step in order to calculate the probability of a sentence is to estimate the probability of a word given the previous n words. For ease of exposition, we will focus on bigram and unigram models for the remaining of this chapter.

2.2.1 Estimation

The simplest and most intuitive way of estimating the probabilities $P(w_i|w_{i-1})$ of a bigram model consists of counting how many times the word w_i occurs in the data given that the previous word was w_{i-1}. Let $c(w_{i-1}^i)$ denote the number of times bigram w_{i-1}, w_i occurs in a given text. Then

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}^i)}{c(w_{i-1})} \tag{2.6}$$

This estimate is called the *Maximum-Likelihood* (ML) estimate because it maximizes the probability of the data: all probability mass is given to events occurring in the data, none to events outside the data.

Consider a training set containing only three sentences

(JOHN LIKES PIZZA, JOE EATS BREAD, HE EATS PIZZA)

Then the probability of the sentence HE EATS PIZZA according to the maximum-likelihood estimator is given by

$$\begin{aligned} P(\text{HE EATS PIZZA}) &= P(\text{HE}|w_0)P(\text{EATS}|\text{HE})P(\text{PIZZA}|\text{EATS})P(w_{n+1}|\text{PIZZA}) \\ &= \frac{1}{3} \times 1 \times \frac{1}{2} \times 1 \approx 0.17 \end{aligned}$$

where w_0 is the beginning-of-sentence marker, making Equation 2.5 meaningful for $i = 1$, and w_{n+1} is the end-of-sentence marker. Markers are important information sources, since some words are more likely to start or end sentences than others. The determiner THE, for example is more likely to start a sentence than to end it. Both markers are added to the product of Equation 2.5.

2.2.2 Sparse data problem

The intuition behind ML estimators is that a data sample randomly drawn from a population should be a fair representative of the real distribution and therefore should contain all frequency information that is needed. Natural language data, however, tend to be sparse in the sense that they follow a Zipfian distribution[2].

The problem cannot be fixed by taking increasing training sets because there will always be newly created words, proper nouns, technical terms that will not be caught and that cannot be properly estimated. As soon as a new sequence of

[2]There are few common words and many uncommon words (Manning and Schütze, 2001).

words shows up, the probability of the corresponding sentence becomes zero. For the set above, the probability of HE LIKES PIZZA is zero, since $P(\text{LIKES}|\text{HE}) = 0$. A speech recognizer with this language model would not recognize this sentence. Thus, for language purposes, the ML estimator is qualitatively wrong.

It is also quantitatively wrong since it will assign non-zero probabilities to events that occur only once. It creates an artificial cut-off between unfrequent events in general. If an unfrequent word (e.g. McGYVER) happens to be present in a text, it does not mean that it is more common than another word (e.g. RAMBO). It is simply a consequence of the topic covered in the text.

Given the limitations of the ML estimation procedure, it is necessary to find an alternative estimator that is able to assign non-zero probabilities to unseen events while keeping reliable information about events that were seen.

2.3 Smoothing

Some techniques have been devised to cope with the sparse data problem. They are called *smoothing* techniques, since they tend to make distributions more "smooth", by adjusting low probabilities upward and high probabilities downward. The underlying idea is to discount counts $r = c(w_{i-n+1}^{i})$ from events w_{i-n+1}^{i} and give them to events that did not occur in the training set. The new counts of w_{i-n+1}^{i} are then given by r^*. The main constraint is that

$$\frac{\sum r^*}{N} = 1 \tag{2.7}$$

where N is the total number of n-grams in a sample. We present next two smoothing techniques. We compare their properties and justify our choice for this thesis. There are several other techniques available. An extensive overview is given by Chen and Goodman (1998).

2.3.1 Additive smoothing

A very simple way of reserving mass for unknown events lies in adding some constant k to every n-gram count in the training set and renormalizing appropriately. Then $r^* = (r+k)\frac{N}{N+kS}$, where S is the total number of n-gram types[3]. For unigram models, for instance, S is the vocabulary size[4]. In the case of $k = 1$, we have the *Add-One* method given by

$$r^* = (r+1)\frac{N}{N+S} \tag{2.8}$$

The sentence HE LIKES PIZZA can now have a smoothed estimate given by

[3] *Types* are n-gram classes; *tokens*, n-gram occurrences. In THE BOY WITH THE HAT, there are 4 unigram types and 5 unigram tokens or 4 bigram types and tokens.

[4] The estimators described here assume a fixed vocabulary. Thus, all words are known, and the problem is estimating probabilities for *sequences of words* that were not seen in the data.

$$\begin{aligned} P(\text{HE LIKES PIZZA}) &= P(\text{HE}|w_0)P(\text{LIKES}|\text{HE})P(\text{PIZZA}|\text{LIKES})P(w_{n+1}|\text{PIZZA}) \\ &= \frac{2}{6} \times \frac{1}{2} \times 1 \times 1 \approx 0.17 \end{aligned}$$

The main advantage of Add-One is simplicity. The main problem is its tendency to overestimate unseen events. The probability given to the unseen sequence HE LIKES PIZZA in an Add-One smoothed distribution is the same as the probability of the seen sequence HE EATS PIZZA in an un-smoothed distribution.

Indeed, Gale and Church (1994) showed that Add-One performs rather poorly when compared to other more sophisticated techniques. They used as a training set a 44-million word 1988 newswire from Associated Press and compared results from the Add-One estimator with results from Good-Turing (see Section 2.3.2) and held-out[5] estimators. They found that Add-One misses the target by a factor ranging from 10 to 3,000.

Moreover, they showed that Add-One would only work if the ratio of unseen n-gram types and observed types equals the ratio of total types and training sample size. Since this equality would only hold by coincidence, the Add-One estimator would only give good estimates in very specific situations.

Finally, they compared error rates between the ML estimator and Add-One and found that Add-One performs even worse. The lesson here is that smoothing with a bad estimator is worse than not smoothing at all.

2.3.2 Good-Turing discounting

A more sophisticated discounting procedure is given by the Good-Turing (GT) estimate. Let $E(X)$ be the expected value of random variable X and N_r be the frequency of the frequency r in the training data (i.e. the number of events with exactly r counts). Then the new count r^* is given by

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

Because of the Zipfian nature of word distributions in texts, it generally holds that $E(N_{r+1}) < E(N_r)$. Therefore, it is usually the case that $r^* < r$. In other words, events that did occur in the training data have their frequencies downgraded. To convert the new counts to probabilities, it is necessary to renormalize them. Let N be the total number of counts in an n-gram distribution and r be the count of n-gram w_{i-n+1}^i. Then

(2.10) $$P_{\mathrm{GT}}(w_{i-n+1}^i) = \frac{r^*}{N}$$

The GT estimator is based on the Good-Turing theorem[6]. The theorem,

[5] In held-out estimation, the training set is split. Estimates are obtained by counting occurrences in the retained split and normalizing by occurrences in the held-out split.

[6] The theorem was proved by Alan Turing and his assistant I.J. Good during their collaboration in the Second World War effort to break German codes at Bletchley Park, UK. The work of this period became known to a wide public through Robert Harris's 1995 thriller "Enigma" and the film of the same name released in 2002, starring Kate Winslet.

obtained in the context of estimating populations of species, was adapted later to linguistics. To derive the formula, it is assumed that the events of the model being smoothed (e.g. words in a text) are generated by a binomial distribution. As a result, a sequence of words is treated as a sequence of independent drawings.

Let us consider a sequence of $m > 100$ words and a unigram distribution. Suppose that 100 words in this sequence appeared just once. So, the ML estimate for each such word would be $p = 1/m$ (the probability of success in a Bernoulli trial[7]). In m trials, the number of words expected to be seen once, according to a binomial distribution defined by p and m, is given by $100 \times \binom{m}{1} p(1-p)^{m-1}$. If $m = 2000$, then we would expect to find 37 words occurring just once. But we did see 100. The problem is that many events with lower and higher probabilities than p can be expected to appear just once.

GT theorem is based on the following idea: let us count the contribution of each event's observed frequency to every possible frequency in the distribution (including frequency 'zero'). In this way, the GT estimator can work out the expected contribution of all other frequencies to $r = 0$. Paraphrasing Gale and Sampson's (1995), the theorem is about the frequency one would expect in an *additional* sample for events which occurred r times in an *observed* sample. A proof of Good-Turing theorem is given in Chen and Goodman (1998).

For language, the binomial assumption is usually false. If the word GREEN is drawn, then the following word is more likely to be a noun than a verb. So, a sequence of words in a sentence is *not* a sequence of independent trials. Despite that, Good-Turing estimation has improved performance in language models.

It is common practice to make $E(N_r) \approx N_r$, that is, to approximate the expected frequency of frequencies $E(N_r)$ by its empirical value N_r. Good-Turing estimator can then be easily computed as

$$r^* = (r+1)\frac{N_{r+1}}{N_r} \tag{2.11}$$

For low r, that is a good approximation, since most events occur very rarely and so there is plenty of statistics for N_r. As r grows, however, the approximation becomes less and less reliable, as the statistics start to vanish. The number of counts reserved for unseen events is $N_0 0^* = N_0 \frac{N_1}{N_0} = N_1$. So the amount of probability mass reserved for unseen events becomes N_1/N.

Note that the total number of counts remains the same:

$$\sum_{r=0}^{r_{max}} r^* N_r = \sum_{r=0}^{r_{max}} (r+1) N_{r+1} = \sum_{r=1}^{r_{max}} r N_r = N \tag{2.12}$$

The discounted mass $\sum_{r=1}^{r_{max}} \frac{(r-r^*)}{N}$ can then be distributed to unseen events.

One issue when using GT estimator is that N_r distributions are very noisy in general. For a given distribution, it might be the case that $N_{201} \neq 0$ and $N_{200} = 0$, yielding an undefined r^* value. In particular, for the most common event, r_{max}, GT estimate will return value 0.

[7]Random process satisfying three assumptions: i) only two possible outcomes (success, failure); ii) independence among trials; iii) for each trial, $P_{\text{success}} = p$, $P_{\text{failure}} = 1 - p$.

The main difference among methods implementing the GT estimator lies in how they decide to smooth the N_r distribution. For this thesis, we followed Gale and Sampson's (2001) Simple-Good Turing (SGT) method. They approximated the N_r distribution with a downward-sloping log-log curve. SGT perfoms as well as alternative smoothing techniques, such as *cross validation*[8], but it is simpler and faster. We describe SGT in detail in Chapter 3.

2.4 Combining estimators

Saving probability mass for unseen events is an important step in order to smooth a distribution. Yet the problem of how much mass should be assigned to which events still remains. The first idea is to divide it uniformly among them. This strategy, however, neglect differences in the events' structure.

Suppose we want to construct a bigram model on a training set where $c(\text{ON-LINE TRADE}) = 0$ and $c(\text{ON-LINE COMMERCE}) = 0$. Then, according to both Add-One and Good-Turing estimators

$$P(\text{TRADE}|\text{ON-LINE}) = P(\text{COMMERCE}|\text{ON-LINE})$$

even though intuitively we should have

$$P(\text{TRADE}|\text{ON-LINE}) > P(\text{COMMERCE}|\text{ON-LINE})$$

since TRADE is presumably used more often than COMMERCE.

One way of taking into account the structure of unseen events is by combining evidence from different information sources, giving higher values to the most reliable source and lower values to the less reliable one. We describe two such methods, namely, linear interpolation, developed by Jelinek and Mercer (1980), and back-off estimation, introduced by Katz (1987). We will also motivate our decision for choosing back-off as a smoothing technique for DOP distributions.

2.4.1 Jelinek-Mercer interpolation

Interpolation has been used to combine estimators. The aim is to assign probability to events according to the distribution with the most reliable information about the event. An indication of reliability is the frequency magnitude. High counts are taken as good evidence for an event; low counts are distrusted of.

In interpolation estimators, no probability mass is explicitly reserved for unseen events. It is only reallocated according to lower-order distributions. The weights will decide how much probability should be given to which events. An n-gram linear interpolated model can be recursively defined as follows:

$$P_{int}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{int}(w_i|w_{i-n+2}^{i-1})$$

[8] In cross validation, retained and held-out splits are alternated and the results, averaged.

To end the recursion, we can take the ML unigram estimator. Hence, for $n = 1$, $P_{int}(w_i) = P_{\mathrm{ML}}(w_i)$. In the example given above, a bigram interpolated model can be defined as:

$$P_{int}(w_i|\text{ON-LINE}) = \lambda_{\text{ON-LINE}} P_{\mathrm{ML}}(w_i|\text{ON-LINE}) + (1 - \lambda_{\text{ON-LINE}}) P_{\mathrm{ML}}(w_i)$$

Since we would have $P_{\mathrm{ML}}(\text{TRADE}) > P_{\mathrm{ML}}(\text{COMMERCE})$, the model would assign $P_{int}(\text{TRADE}|\text{ON-LINE}) > P_{int}(\text{COMMERCE}|\text{ON-LINE})$, as desired. The parameter $\lambda_{\text{ON-LINE}}$ and the unigram probabilities (lower-order distribution) decide how much probability is to be given to each event.

One issue of interpolation models is the computation of the parameters $\lambda_{w_{i-n+1}^{i-1}}$. Fortunately, there is an efficient hill-climbing algorithm (Baum, 1972) that searches for the $\lambda_{w_{i-n+1}^{i-1}}$ that maximizes the probability of the data. To obtain meaningful results, it is necessary to train the parameters in a set different than the set where the probabilities P_{ML} were estimated[9].

The computation of parameters λ can be very time-consuming, since the number of possible histories in any statistically significant sample is large. A frequent strategy is to collapse or to *bucket* histories that are believed to be related. Putting together all histories with the same frequency has been shown to give good results (Chen and Goodman, 1998).

For DOP distributions, the computation of λ tends to be very costly. Consider the Noun-Phrase THE BOY LESSONS (see Figure 1.5 in Chapter 1). In a string-based trigram model, there is only one parameter to be computed, namely, $\lambda_{\text{THE BOY}}$. In a DOP-based model, the are at least two, since there are two different subtree structures, i.e. two histories, yielding the same string. Thus, using Jelinek-Mercer interpolation to smooth a DOP grammar would be computationally very expensive. Bucketing would not help much as well, since it is not immediately clear how to group subtrees into meaningful classes.

In the following, we present another technique used to both smooth and combine information for estimators that is computationally less expensive and, therefore, more suitable for DOP-like grammars.

2.4.2 Katz back-off

Katz (1987) developed a technique in which events in a higher-order distributions are discounted only if their counts are less than a fixed cut-off value. The discounted mass is then distributed to unseen events by means of lower-order distributions. For a trigram distribution defined on MODERN GREEN, if the next input word is CAR and $c(\text{MODERN GREEN CAR}) = 0$, then Katz suggest to "back-off" to the next lower-order distribution – a bigram distribution defined on GREEN –, where possibly $c(\text{GREEN CAR}) > 0$.

A back-off distribution can be recursively defined as follows:

[9] If both are estimated on the same data, then the optimal result is the one that sets $\lambda_{w_{i-n+1}^{i-1}} = 1$ for all histories w_{i-n+1}^{i-1}, in which case we are back to the ML estimator.

$$P_{bo}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} P'_{\mathrm{GT}}(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i-1}) > 0, \\ \alpha(w_{i-n+1}^{i-1}) \times P_{bo}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \end{cases}$$

where P'_{GT} is a Good-Turing-type function; $\alpha(w_{i-n+1}^{i-1})$, a normalization constant and P_{bo} is defined in the same way as the back-off function above. To end the recursion, as with Jelinek-Mercer interpolation, we can take the ML unigram estimate. So, when $n = 1$, $P_{bo}(w_i) = P_{\mathrm{ML}}(w_i)$. The normalization constant depends on the event's history and is given by:

$$\alpha(w_{i-n+1}^{i-1}) = \frac{1 - \sum_{w_i : c(w_{i-n+1}^{i}) > 0} P'_{\mathrm{GT}}(w_i|w_{i-n+1}^{i-1})}{1 - \sum_{w_i : c(w_{i-n+2}^{i}) > 0} P_{bo}(w_i|w_{i-n+2}^{i-1})}$$

When $c(w_{i-n+1}^{i-1}) = 0$, define $P_{bo}(w_i|w_{i-n+1}^{i-1}) = P_{bo}(w_i|w_{i-n+2}^{i-1})$.

The Good-Turing function here differs from Equation 2.10 because it is only applied to events whose counts are smaller or equal to a fixed cut-off value. In its pure form, the GT estimator discounts from events with any frequency r ($r \in \mathbb{N}$). Let $N = \sum_{r=1}^{r_{max}} r\, N_r$. For an n-gram w_{i-n+1}^{i} with count $r = c(w_{i-n+1}^{i})$, P'_{GT} can be defined as:

$$P'_{\mathrm{GT}}(w_i|w_{i-n+1}^{i-1}) = d_k(r)\frac{r}{N}$$

where $d_k(r)$ is a *discount ratio* that depends on the cut-off value k. In Good-Turing estimation, $d_k(r) = d(r) = \frac{r^*}{r}$ (i.e., in its original formulation, $k = \infty$). In the back-off case, events with more than k counts are considered reliable and are not discounted. So, for $r > k$, $d_k(r) = 1$.

The discount function is chosen such that the resulting discounts are proportional to the discounts predicted by the Good-Turing estimator. Moreover, the total number of counts discounted in the global n-gram distribution has to be equal to the total number of counts that is going to be assigned to n-grams with zero counts. The former constraint implies that

$$1 - d_k(r) = \mu(1 - \frac{r^*}{r})$$

for some constant μ and for $1 \leq r \leq k$. According to the Good-Turing estimate, the number of counts that should be assigned to n-grams with zero counts is N_1 (see Section 2.3.2). Thus, the second constraint implies that

$$\sum_{r=1}^{k}(1 - d_k(r))\, r\, N_r = N_1$$

The unique solution that satisfies both constraints is given by

$$d_k(r) = \frac{\frac{r^*}{r} - \frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}$$

Note that when $d_k(r) = 1$, $P'_{\mathrm{GT}} = P_{\mathrm{ML}}$. Katz suggest using $k = 5$. The advantage of using a cut-off value for discounting is that generally one does not need to smooth the N_r distribution. For low r values and large sample sizes, the distribution tends to be less noisy. The back-off procedure implemented in this thesis, however, sets $k = \infty$ (Simple Good-Turing, see Chapter 3).

Let us illustrate the back-off estimation procedure. Assume a bigram distribution on the word GREEN and assume that the bigrams GREEN CUT, GREEN CAT and GREEN CAR are not found in this distribution. Let the total bigram probability mass estimated by P'_{GT} be 0.97. Then, the reserved mass for unseen events is 0.03. If $P(\text{CUT}) = 0.1$, $P(\text{CAT}) = 0.2$ and $P(\text{CAR}) = 0.3$, then the back-off estimates of those events will be given by:

$$
\begin{aligned}
P_{bo}(\mathrm{CUT}|\mathrm{GREEN}) &= 0.03 \times \frac{P_{\mathrm{ML}}(\mathrm{CUT})}{P_{\mathrm{ML}}(\mathrm{CUT}) + P_{\mathrm{ML}}(\mathrm{CAT}) + P_{\mathrm{ML}}(\mathrm{CAR})} &= 0.005 \\
P_{bo}(\mathrm{CAT}|\mathrm{GREEN}) &= 0.03 \times \frac{P_{\mathrm{ML}}(\mathrm{CAT})}{P_{\mathrm{ML}}(\mathrm{CUT}) + P_{\mathrm{ML}}(\mathrm{CAT}) + P_{\mathrm{ML}}(\mathrm{CAR})} &= 0.010 \\
P_{bo}(\mathrm{CAR}|\mathrm{GREEN}) &= 0.03 \times \frac{P_{\mathrm{ML}}(\mathrm{CAR})}{P_{\mathrm{ML}}(\mathrm{CUT}) + P_{\mathrm{ML}}(\mathrm{CAT}) + P_{\mathrm{ML}}(\mathrm{CAR})} &= 0.015
\end{aligned}
$$

As with Jelinek-Mercer interpolation, Katz back-off is able to capture differences in events' internal structure, by taking into account information from lower-order distributions.

A back-off model can be formulated as a non-linear interpolation:

$$P_{bo}(w_i|w_{i-n+1}^{i-1}) = \lambda(w_{i-n+1}^{i})P'_{\mathrm{GT}}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda(w_{i-n+1}^{i}))\alpha(w_{i-n+1}^{i-1})P_{bo}(w_i|w_{i-n+2}^{i-1})$$

where the weights λ can be defined for every n-gram w_{i-n+1}^{i} as

$$\lambda(w_{i-n+1}^{i}) = \begin{cases} 1 & \text{if } c(w_{i-n+1}^{i}) > 0, \\ 0 & \text{otherwise} \end{cases}$$

The major differences between Jelinek-Mercer interpolation and Katz back-off is that the former discounts probabilities from all events in the higher-order distribution, whereas the latter discounts only from events whose counts are considered unreliable. Another difference is that back-off models explicitly reserve probability mass to unseen events and do so by means of Good-Turing estimation. Jelinek-Mercer model, on the other hand, saves probability indirectly, as a consequence of training the parameters λ.

Both linear interpolated and back-off models suffer from being unable to identify "grammatical zeroes". A bigram model can be seen as a two-dimensional matrix where the rows and columns contain all the word types appearing in the text. For each row, a distribution can be defined. Each distribution will present several zeroes. Some of them follow from the sparse data problem, but there are some zeroes which should really be there. Events such as THE THE are very unlikely and could only happen as a repetition error in speech. In this case, we would like to have $P(\text{THE}|\text{THE}) \approx 0$. Nevertheless, both interpolated and back-off bigram models would assign a high probability to this event when compared to other unseen events, since the unigram probability $P_{\text{ML}}(\text{THE})$ is high.

Grammatical zeroes take probability mass that could, otherwise, be better distributed among other unseen events. Alternative smoothing techniques, such as Kneser-Ney smoothing, are able to tackle this kind of problem.

All in all, back-off models present a fast and accurate way of smoothing distributions. In the next chapter, we outline a general back-off model for STSG grammars that is inspired by Katz approach.

Chapter 3

Back-off as parameter estimation for DOP models

In this chapter, we present a general framework for a Back-Off DOP model. We formalize the concept of 'back-off' subtree and describe an instantiation of such a model using the particular case where back-off subtrees differ from the original ones by a single lexical item. An algorithm implementing the idea is presented and the estimation procedure used here is compared with related work.

3.1 Outline of a general model

One of the fundamental assumptions underlying DOP models is that a treebank encodes a hidden STSG. The goal of DOP modelling is then to find the parameters that best represent this STSG. Annotated trees, however, are the only evidence available for such models. Thus, we need to find a way of capturing as much information from the hidden relations in these trees as we can.

Each parse tree in a corpus encodes several derivations and these derivations are interrelated, since the subtrees involved in these derivations are interrelated. Whenever a subtree t is extracted from a corpus, a subtree t' – smaller than t and 'contained' in it – is also extracted. Thus, the subtree extraction procedure populates the subtree space with several pairs t and t' that were related in the past. This means that information about derivations and their past relation with the treebank trees is also encoded in the subtree distributions.

Consider a toy treebank as illustrated in Figure 3.1(a–b). The derivations[1] represented by Figures 3.1(a,c) yield both the same parse tree. In addition, they yield a parse tree that was previously seen in the treebank. More importantly, the subtrees participating in these derivations originated from the same treebank parse tree. This is contrast with Figure 3.1(d), which represents a derivation of a

[1]The graph in Figure 3.1(a) can represent three distinct entities: i) a parse tree; ii) a derivation; iii) a subtree. The correct interpretation should be clear from the context.

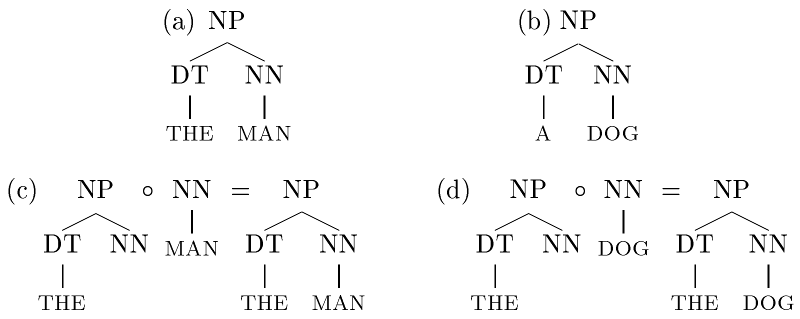

Figure 3.1: Derivations of THE MAN(c) and THE DOG(d) given a treebank (a–b). The derivation in (c) is a back-off derivation of (a); (d) is a derivation not previously observed in the treebank, obtained by using a back-off subtree of (a).

tree that was not previously seen in the corpus. Clearly, the subtrees involved in that derivation did not co-occurr in the treebank. Nevertheless, the derivation is possible because it makes use of a smaller fragment ('back-off') obtained from 3.1(a) during the extraction procedure.

The derivations in 3.1(a,c) are interrelated, since the subtrees participating in the derivations are also interrelated. Current DOP models, however, treat subtree events as if they were disjoint. Some of the dependencies between the events may be recovered during parsing, by summing up over all possible derivations, in order to find the Most Probable Parse for an input string. In practice, however, only the Most Probable Derivation is considered during disambiguation. Because the MPD does not take into account all possible combinations of fragments, it loses this past information about joint dependencies.

We would like to come up with an estimator that is able to tie up subtree estimates such that the joint dependencies experienced in the past can be partially recovered even when we are not able to sum up over all possible derivations. One way of doing that is by treating subtrees events as back-offs of each other. The resulting derivations would be tied up in the following sense: if a derivation using subtree t is not available, then use a derivation involving subtree t' that is a back-off of t. This procedure can be compared to Katz smoothing technique (see Chapter 2, Section 4.2). In that case, the counts of t would be discounted and reserved for the case when a derivation with t' is needed. A normalization factor α dependent on the previous context would assure the properness[2] of the resulting distribution.

If t, t' and t'' are subtrees in an STSG G' associated with a given DOP model and $t' \circ t'' = t$, then a Back-Off DOP model could be defined as

$$(3.1) \qquad P_{bo}(t|X) = \begin{cases} P'_{\mathrm{GT}}(t|X) & \text{if } c(t) > 0, \\ \alpha(X, t')\ P(t'|X)\ P(t''|Y) & \text{otherwise} \end{cases}$$

[2]A probability distribution defined on an event space Ω is *proper* if $\sum_{x \in \Omega} P(x) = 1$.

where Y is the root node label of t'', P'_{GT} is a Good-Turing-type function and $c(t)$ denotes the counts of subtree t in the X-rooted subtree distribution. In the derivations of Figure 3.1(c,d), t' is the NP-rooted subtree starting the derivation, t'' is the NN-rooted preterminal tree, X is NP and Y is NN.

It is important to notice that what we mean by 'back-off' here is different than its usual meaning in language modelling. There, backing off helps the system to handle unseen events, by reserving some probability mass for them. Here, unseen events can be handled already without the need of any smoothing technique. Thus, back-off is not used here to make the model more robust. Our main goal is to use the tools provided by the back-off technique to *transfer probability mass* across derivations, so that we can tie them up. By doing so, we end up tackling some of the shortcomings of previous DOP models, including the bias of DOP1 towards large subtrees and the assumptions that derivations are equally likely and independent.

Note that Equation 3.1 treats a derivation step $t' \circ t'' = t$ as a multiplication of subtree probabilities. That is called the *independence assumption* and has been used by all DOP models so far. It is given by

$$\begin{aligned} P(t|X) &= P(t', t''|X) \\ &= P(t'|X)\, P(t''|X, t') \\ &\approx P(t'|X)\, P(t''|Y) \end{aligned} \tag{3.2}$$

The most important difference in our model, however, is that the probabilities involved in the multiplications are changed in a way that tries to interrelate the subtrees involved in the derivations, providing more sensitive estimates.

In the following we define formally the concepts of back-off subtree and back-off derivation and describe the simplified version implemented in this thesis.

3.1.1 Definitions

We want to back-off gradually across derivations without big leaps in a way that resembles back-off models of the language modelling tradition. In those cases, an n-gram is backed-off to an (n-1)-gram and the process follows in a stepwise manner. The equivalent of 'word' in subtree distributions are depth-1 subtrees. Thus, we will use them as steps in our proposed model.

Definition 3.1 *Let $G' = \langle V'_N, V'_T, R', S, P' \rangle$ be an STSG associated with a given DOP model. A subtree $t' \in R'$ is called a* <u>back-off</u> *of another subtree $t \in R'$ if there exists a depth-1 subtree $t'' \in R'$ such that $t' \circ t'' = t$.*

Thus, two trees are back-offs of each other if they differ by a depth-1 subtree substitution. A depth-1 subtree t is *unlexicalized* if its yield contains no lexical items. A depth-1 subtree t is called a *preterminal* subtree if $N(t) = 0$ and $y(t) \in V'_T$ (i.e. preterminal subtrees contain only a part-of-speech tag and a word). Back-off subtrees can be grouped into two classes:

- A subtree $t' \in R'$ is called *structural back-off* of another subtree $t \in R'$ if there exists an unlexicalized depth-1 subtree $t'' \in R'$ such that $t' \circ t'' = t$;
- A subtree $t' \in R'$ is called a *lexical back-off* of another subtree $t \in R'$ if there exists a depth-1 preterminal subtree $t'' \in R'$ such that $t' \circ t'' = t$.

The definitions above can be extended to derivations[3].

Definition 3.2 *Let d and d' be two derivations of the same parse tree and let subtree $t' \in R'$ participate in d'. Derivation d' is called a* back-off derivation *of d if there exists a subtree t participating in d such that t' is a back-off of t.*

Note that a derivation is a back-off of another derivation only if both yield the same parse tree. Figure 3.1(a) and (d) show two derivations. The latter starts with a subtree which is a back-off of the subtree represented by 3.1(a). The derivation in which the back-off subtree participates, however, is *not* a back-off derivation of 3.1(a), since they result in different parse trees.

Back-off derivations can be classified according to the subtrees they involve. If a structural back-off is involved, the derivation is called *structural back-off derivation*; if a lexical back-off is involved, it is a *lexical back-off derivation*.

Figure 3.2 illustrates a series of back-off derivations of a given tree. Note that the probability mass flows from shorter to longer derivations, creating an ordering in the derivation space. The derivations in the first four tiers are back-off derivations of the A-rooted subtree; the last two tiers show back-off derivations of the C′-rooted subtree. In this example, there are only two structural back-offs; larger parse trees will allow for more of them. For convenience, we restrict ourselves in this thesis to lexical back-offs and lexical back-off derivations.

3.2 Backing-off with lexical items

As with n-grams, we want our model to back off from n-word sequences to (n-1)-word sequences. A sequence here is a subtree. We call a subtree a *n-word subtree* if it has exactly n lexical items in its frontier. A constituent is a special case of a n-word subtree that has n lexical items in its frontier and no substitution sites. Similarly to n-gram models, we want our model to back off from n-word subtrees to (n-1)-word subtrees.

For expository reasons, we will treat a subtree distribution defined on a root node X as a multiset[4]. A *multiset* is a collection of elements. Contrary to *sets*, multisets allow multiple occurrences of the same element. Thus, corpora and subtree distributions can be represented as multisets, since frequency information is preserved. In this work, we will consider only finite multisets.

[3]The definitions of structural and lexical back-offs assume that the utterance-analyses distinguish between non-terminals and preterminals (POS tags): all depth-1 subtrees are either lexicalized or not lexicalized. This assures complete coverage of depth-1 subtree space.

[4]We decided to use multisets so that we could talk about 'partitioning a subtree distribution' in a formal setting without losing the useful distinction between types and tokens.

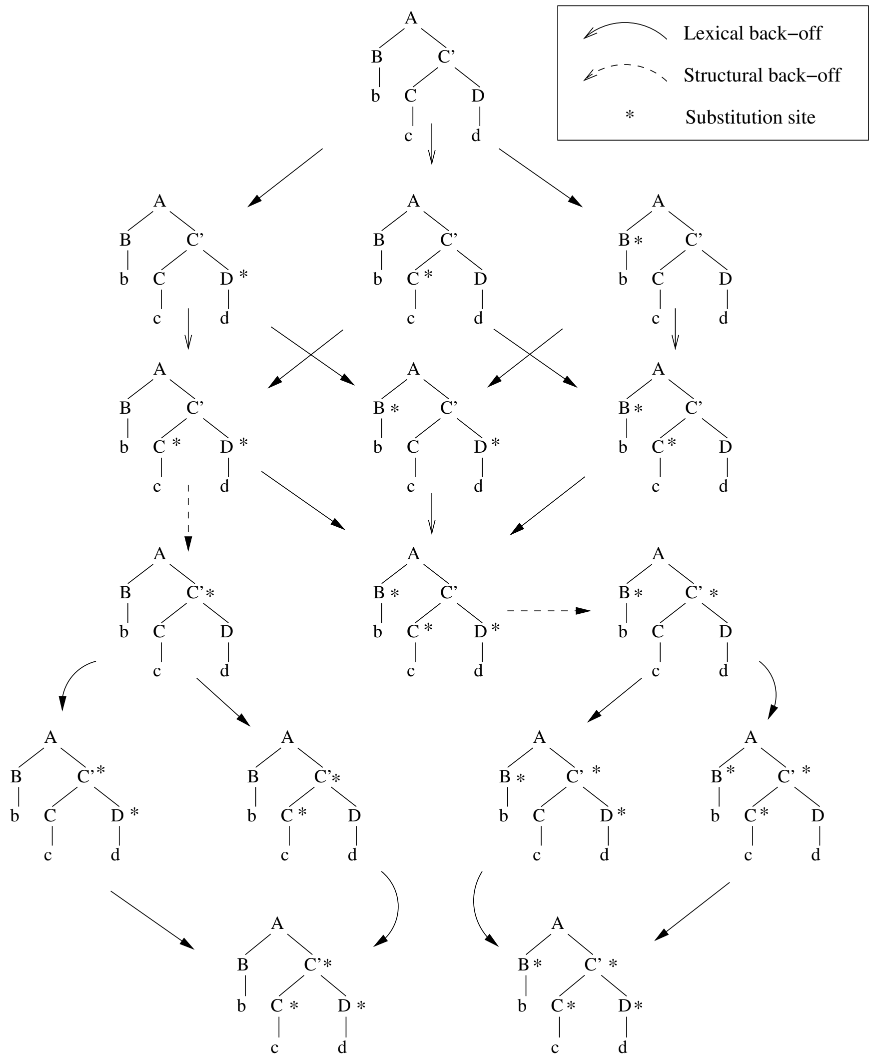

Figure 3.2: Flow of probability mass across back-off derivations of the same parse tree. Derivations are uniquely identified by a tree decorated with '*', which represents substitution sites. Back-off derivations can be lexical and structural. In this thesis, we focus on lexical back-off derivations.

The goal of this section is to describe an algorithm that updates the probability estimates of DOP models with back-off estimates. In the following, we define some basic concepts of set theory adapted to multisets and formalize our treatment of subtree distributions in order to make the algorithm clear.

3.2.1 Partitioning subtree space

A subtree *token* is an element in a multiset. Let $\in$ denote *multiset membership*. A multiset A is a submultiset of a universal multiset U (denoted as $A \subset U$), if for all $t \in A$, we also have $t \in U$. Let A, $B \subset U$. The *union* of multisets A and B ($A \cup B$) is a multiset containing $t \in U$ such that if t occurs l times in A and m times in B, then t occurs $l + m$ times in $A \cup B$. If $A = \{t_1, t_1, t_2\}$ and $B = \{t_1, t_2, t_3\}$, then $A \cup B = \{t_1, t_1, t_1, t_2, t_2, t_3\}$. The *intersection* of multisets A and B ($A \cap B$) is a multiset consisting of $t \in U$ such that if t occurs l times in A and m times in B, then t occurs $min(l, m)$ times in $A \cap B$. In the example above, $A \cap B = \{t_1, t_2\}$. A multiset is *empty* ($\varnothing$) if it contains no elements. Two multisets A and B are *disjoint* if $A \cap B = \varnothing$. A *partition* of multiset A is a set Q containing n+1 non-empty disjoint submultisets A_i of A such that $\cup_{i=0}^{n} A_i = A$.

Let U^X be the multiset containing all subtrees whose root node is X. We want to partition the X-rooted subtree distribution so that subtrees with the same number of lexical items belong to the same submultiset. Let $Q_n^X = \{A_0^X, \ldots, A_n^X\}$ be a partition of U^X containing n+1 disjoint submultisets A_i^X. Each A_i^X represents a collection of subtrees containing exactly i lexical items.

We want to transfer probability mass from subtrees with n words to their corresponding back-offs with n-1 words. Partitioning the subtree space in this way, however, does not allow us to capture a fundamental difference among subtrees with the same number of lexical items: some of them are not back-offs.

Consider a partition Q_2^{NP} and submultisets $A_1^{NP}, A_2^{NP} \subset Q_2^{NP}$. It is possible to have a subtree $t' \in A_1^{NP}$ for which there is no $t'' \in R'$ such that $t' \circ t'' = t$ for some $t \in A_2^{NP}$. In other words, it is possible to have a subtree $t' \in A_1^{NP}$ that is not a lexical back-off of any subtree $t \in A_2^{NP}$. This is the case with lexicalized unary subtrees. Each non-terminal node of an unary subtree has only one daughter. A lexicalized unary subtree has a lexical item in its frontier (it is a constituent). These subtrees could not be the result of a back-off of some 2-word subtree in A_2^{NP}, since this would result in a subtree that is not unary. 0-word unary subtrees, however, may be back-offs of lexicalized unary subtrees. In short, i-word constituents are not back-offs of (i+1)-word subtrees but (i-1)-subtrees may be back-offs of i-word constituents. In general, a subtree *cannot* be a lexical back-off if it has no part-of-speech tags in its frontier.

In order to capture these differences, we need to define another partition on U^X that is able to distinguish between back-off and non-back-off subtrees. Let $R^X = \{A_{bo}^X, A_{\overline{bo}}^X\}$ be a partition of U^X containing 2 disjoint submultisets representing collections of back-off and non-back-off subtrees.

Now we can refine the subtree space. Let $A_{i,bo}^X = A_i^X \cap A_{bo}^X$ represent a submultiset containing subtrees with the following properties: a) each one has i lexical items; b) each one is a back-off of some subtree in A_{i+1}^X. For $i = n$, let $A_{n,bo}^X = A_n^X$ (in this case, it does not matter whether subtrees are back-offs or not, since there are no subtrees with more than n lexical items).

Borrowing from n-gram terminology, we will refer to submultisets $A_i^X, A_{i+1}^X \subset Q_n^X$ in the following way: A_{i+1}^X is said to be a *higher-order distribution* with respect to A_i^X and A_i^X is a *lower-order distribution* with respect to A_{i+1}^X.

3.2.2 Correction factor α

If we want to change probabilities of derivations we have to change the probabilities of the subtrees involved in these derivations. So, we work at the level of subtree distributions, not at the level of derivation distributions.

In order to transfer probability mass across back-off derivations, we could implement the following procedure: for each subtree t in a submultiset A_i^X in a partition Q_n^X, discount the counts of t, using a Good-Turing-like function, and give these counts to the corresponding back-off subtree $t' \in A_{i-1,bo}^X$. Nevertheless, restricting the discounting to a distribution conditioned on an entire subtree is unpractical, given the extreme scarcity of data for such distributions. **Appendix A** shows how such a back-off procedure would look like.

In this work, instead, we lump together all subtrees in the same submultiset A_i^X and give the discounted counts obtained from these subtrees to all subtrees in the next lower-order distribution that contain possible back-offs (i.e. $A_{i-1,bo}^X$). The probability mass discounted from subtrees in submultiset A_i^X is given by

$$\sum_{t:t\, \in\, types(A_i^X)} P(t|X) - P_{\mathrm{SGT}}(t|X) \tag{3.3}$$

where *types*(A_i^X) is the set of subtree *types* in multiset A_i^X. This assures that $\sum_t P(t|X) = 1$ for all types[5] t occurring in A_i^X. $P(t|X)$ is the maximum-likelihood estimator (see Chapter 2, Section 2) and $P_{\mathrm{SGT}}(t|X)$ is the Simple Good-Turing estimator, described in the next section.

According to the independence assumption, the probability of the next derivation in a back-off hierarchy starting with $t \in A_i^X$ is $P(t'|X)\,P(t''|Y)$, where $t' \in A_{i-1,bo}^X$ and $t'' \in R'$ are such that $t' \circ t'' = t$[6]. For all $t \in A_i^X$, the total probability mass of the corresponding back-off derivations is given by

$$\sum_{t',t'':t' \in\, types(A_{i-1,bo}^X),\, t'' \in R'} P(t'|X)\,P(t''|Y) \tag{3.4}$$

where Y is a function of t'. If we distributed the reserved mass only among these derivations, no smoothing effect would be accomplished, since no *new* derivation would get any of the reserved mass. The derivation illustrated in Figure 3.1(d), for example, is not a back-off derivation of any higher-order derivation, since there is no subtree in the higher-order distribution (Figure 3.1(a–b)) whose yield is THE DOG. By dividing the reserved mass only among back-off derivations, we would not make room in the derivation space for that new derivation.

[5]We use the symbol t to refer to both subtree types and tokens. The correct meaning should be clear from the context.

[6]As assumed before, t'' is a preterminal subtree and R' is the set of partial-parse trees associated with a given STSG.

To add a smoothing side to this probability transfer procedure, we take all preterminal subtrees $t'' \in R'$. In other words, we allow all possible combinations of t' and t'', provided that they yield a parse. Since $\sum_{t'' \in R' : r(t'')=Y} P(t''|Y) = 1$, the resulting probability mass of the lower-order distribution becomes

$$\sum_{t' : t' \in \, types(A^X_{i-1,bo})} P(t'|X) \tag{3.5}$$

The correction factor α is obtained by dividing 3.3 by 3.5, yielding

$$\alpha(X,i) = \frac{\sum_{t : t \in \, types(A^X_i)} [P(t|X) - P_{\mathrm{SGT}}(t|X)]}{\sum_{t' : t' \in \, types(A^X_{i-1,bo})} P(t'|X)} \tag{3.6}$$

The correction factor can be interpreted as follows: take the reserved mass from a higher-order distribution, corresponding to a higher-order derivation, and distribute it among all possible derivations involving the next lower-order back-off subtrees. Note that $\alpha(X,i)$ recovers some sensitivity lost by the independence assumption, since it has different values depending on the level i that it belongs to in the hierarchy. For a single distribution U^X, there are many $\alpha(X,i)$ which are related to structural properties of the subtrees involved in the derivations.

Nonetheless, this factor simply corresponds to an average value in the probability transfer process. A *single* subtree in a higher-order distribution will contribute to the probability mass given to *all* subtrees in the lower-order distribution. The transfer is not as tight as it could be.

Every subtree in a lower-order distribution $A^X_{i-1,bo}$ will get a share of the probability taken from the higher-order distribution A^X_i. For $t' \in A^X_{i-1,bo}$, this share is given by $\alpha(X,i)\,P(t'|X)$. Subtree t', however, already has its own share of the whole space given by the maximum-likelihood estimate $P(t'|X)$. Thus, the net result of discounting from A^X_i and giving to $A^X_{i-1,bo}$ is that the share possessed by subtree t' increased $\alpha(X,i)$ times. The updated probability estimate of t' then becomes $(1+\alpha(X,i))\,P(t'|X)$.

The properness of the resulting distribution is guaranteed by the fact that probability mass discounted at one level i is completely absorbed by subtrees in the next level i-1. The relative size of $A^X_{i-1,bo}$ in distribution U^X was enlarged from $\sum_{t'} P(t'|X)$ to $\sum_{t'} (1+\alpha(X,i))\,P(t'|X)$. This corresponds to

$$\sum_{t'} P(t'|X) + \alpha(X,i) \sum_{t'} P(t'|X) = \sum_{t'} P(t'|X) + \sum_{t} P(t|X) - P_{\mathrm{SGT}}(t|X)$$

So, $A^X_{i-1,bo}$ took exactly what was discounted from A^X_i. This is valid for all $i = n, \ldots, 1$ and assures that the 'smoothed' distribution is proper.

3.2.3 Simple Good-Turing

The Good-Turing estimator is implemented using the Simple Good-Turing technique, introduced by Gale and Sampson (2001). The main idea is to smooth the distribution N_r of frequency of frequencies and use this smoothed distribution to calculate the Good-Turing estimator r^*. Following n-gram terminology, we will represent the number of counts of subtree t as $r = c(t)$.

Because of the Zipfian nature of language distributions (n-grams, subtrees), most events will have low frequency. So, there will be several events whose N_r value is 1. However, N_r distributions are not able to distinguish between events that have same frequency of frequencies but different frequencies. Intuitively, an event that occurs 10,000 times should not be classified in the same way as an event that occurs 5,000 ($N_{10,000} = N_{5,000} = 1$). One way of capturing these differences is by averaging each event with the values of its two non-zero neighbors. Gale and Sampson suggest replacing N_r by $Z_r = 2\frac{N_r}{r_n - r_p}$, where r_n represent the *next* non-zero count relative to r and r_p, the *previous* non-zero count. For $r = 1$, let $r_p = 0$; for $r = r_{max}$, let $r_n = 2r - r_p$. This new distribution represents the density of N_r for large r. Note that for small r, $N_r = Z_r$, since the intervals between two consecutive non-zero counts is unity.

The problem is to find a smoothing function that can fill in the gaps inherited from N_r by the Z_r distribution. This function should satisfy two constraints: i) $r^* < r$, for all r; ii) $\frac{r^*}{r} \to 1$ as r increases. The former guarantees that we are taking probability from observed events and making room for new events; the latter comes from the view that high r are more reliable than low ones and, therefore, that their discount rate should be smaller. A very simple and easily computable function that satisfy both constraints is a downward-sloping log-log curve, given by

$$log(Z_r) = a + b\; log(r) \tag{3.7}$$

where a and b are calculated by simple linear regression. For base 10 logarithms, the SGT smoothed proxy for N_r is given by

$$S_r = antilog(a + b\; log(r)) = 10^a\; r^b \tag{3.8}$$

where $b < -1$ in order to satisfy $r^* < r$. Thus, the Good-Turing estimator can be computed in two ways:

$$r^*_{N_r} = (r+1)\frac{N_{r+1}}{N_r} \tag{3.9}$$

$$r^*_{S_r} = (r+1)\frac{S_{r+1}}{S_r} \tag{3.10}$$

For small r, using $r^*_{N_r}$ is better than using its smoothed estimate $r^*_{S_r}$, since N_r values are good approximations for $E(N_r)$ (see Chapter 2). For large r, the opposite holds. So, we need a way of switching between the two estimation procedures. Gale and Sampson suggest using the variance of $r^*_{N_r}$ as an indicator. If the difference between the estimate with N_r and the estimate with S_r exceeds 1.96 times the standard deviation of $r^*_{N_r}$, then use $r^*_{N_r}$; else, take $r^*_{S_r}$[7]. Once $r^*_{S_r}$ is used for the first time, then continue to use it.

[7]This assures that the difference between them is significant with 95% of confidence.

The standard deviation for $r^*_{N_r}$ is approximated[8] with

$$s(r^*_{N_r}) \approx \sqrt{(r+1)^2 \, \frac{N_{r+1}}{N_r^2} \, (1 + \frac{N_{r+1}}{N_r})} \tag{3.11}$$

Using the definitions in Equations 3.9 and 3.10 and the approximation in Equation 3.11, the Good-Turing estimator r^* using the Simple Good-Turing method is given by

$$r^* = \begin{cases} r^*_{N_r} & \text{if } |r^*_{N_r} - r^*_{S_r}| > 1.96 \times s(r^*_{N_r}), \\ r^*_{S_r} & \text{otherwise} \end{cases} \tag{3.12}$$

Thus, the Simple Good-Turing probability estimate is given by

$$P_{\text{SGT}}(t|X) = \frac{r^*}{N'} \tag{3.13}$$

where $r = c(t)$, $N' = \sum Z_r r^*$ and r^* is defined as in Equation 3.12.

Using SGT changes the BO-DOP estimation procedure in two ways when compared to Katz back-off: i) $N' = \sum Z_r r^* \neq \sum N_r r = N$; ii) $k = \infty$. The former follows from the fact that r^* is estimated from the distribution Z_r and not from N_r. So, in order to correctly normalize the probabilities, we need to divide r^* by N' and not by N. This differs from the traditional Good-Turing method and from Katz back-off, since in those cases $N' = N$. The latter is a consequence of the SGT property that $\frac{r^*}{r} \to 1$ as r increases. Because of that, we do not need the parameter k as a cut-off to the back-off estimation. The reason for using k was the intuition that high counts would be more reliable. SGT takes this into account already. In any case, different distributions would require different k values and computing them would be unnecessarily expensive.

3.2.4 BO-DOP estimator

The estimation procedure of Back-Off DOP can now be properly defined.

Definition 3.3 *Let U^X be a multiset representing a subtree distribution defined on root label X. Let $Q^X_n = \{A^X_0, \ldots, A^X_n\}$ and $R^X = \{A^X_{bo}, A^X_{\overline{bo}}\}$ be partitions of U^X and $A^X_{i-1,bo} = A^X_{i-1} \cap A^X_{bo}$ be a submultiset containing subtrees which: a) have i-1 lexical items; b) are back-offs of subtrees in A^X_i. If D is a DOP model, then the corresponding* <u>Back-Off DOP estimator</u> *of D is defined iteratively as:*

for $i = 1, \ldots, n$*:*
  $P(t|X) = P_{SGT}(t|X),\ \forall t \in \mathit{types}(A^X_i)$
  $P(t|X) = (1 + \alpha(X,i)) \times P(t|X),\ \forall t \in \mathit{types}(A^X_{i-1,bo})$

[8]The assumptions behind the approximations are that N_r and N_{r+1} are independent and that $s(N_r) \approx N_r$. The independence assumption is reasonable given the noisy nature of the N_r distribution. The variance approximation works for binomial distributions with low probabilities and this is consistent with Good-Turing assumptions.

where n is the maximal number of lexical items in the subtrees in distribution U^X, P_{SGT} is defined as in Equation 3.13 and $\alpha(X,i)$ is given by Equation 3.6.

In short, a Back-Off DOP estimator can be obtained by splitting a distribution U^X into two 'orthogonal' partitions and estimating stepwisely the parameters α and r^* with subtrees coming from the adequate splits.

3.2.5 Algorithm

The Back-Off DOP estimation procedure is defined as a *bottom-up* iteration to avoid fractionary counts. Good-Turing estimators are only defined for $r \in \mathbb{N}$. Had we gone top-down, the next round in the iteration would find fractionary counts and it would not be possible to do the discounting in a principled manner.

We wish to estimate the back-off probabilities of all possible distributions U^X defined by an STSG. It might be the case that some distributions have subtrees with a maximum number of lexical items $n^X_{max} < n$. Thus, for each distribution, this value has to be found and set.

The pseudo-code for the back-off smoothing algorithm is given by:

INPUT: STSG with probabilities estimated by some DOP model
OUTPUT: STSG with BO-DOP probabilities

for each distribution U^X in the STSG, **do**:
    set $n = n^X_{max}$
    create partition Q^X_n
    create partition R^X
    for $i = 1, \ldots, n$, **do**:
        for t in $types(A^X_i)$, **do**:
            set $P(t|X) = P_{\mathrm{SGT}}(t|X)$
        calculate $\alpha(X,i)$
        for t in $types(A_{i-1,bo})$, **do**:
            set $P(t|X) = (1 + \alpha(X,i)) \times P(t|X)$

These values are pre-computed and inserted in the DOP model's STSG. The efficiency of the method comes from the fact that it is linear in STSG size and it is off-line (prior to the disambiguation phase).

The algorithm has worst-case complexity given by $O(|R'| \times |types(U^X_*)|)$, where R' is the rule set associated with the input STSG and U^X_* is the distribution in the STSG with the largest number of subtree types. The algorithm was implemented in Python[9].

[9]Python is an object-oriented interpreted scripting language created by former University of Amsterdam student Guido van Rossum in the early 1990s while working at the CWI (Center voor Wiskunde en Informatica), in Amsterdam. Its main strengths are clear syntax and structure. Python recently started to be used as an educational and research tool in Natural Language Processing (see Natural Language Toolkit, http://nltk.sourceforge.net).

Figure 3.3 illustrates the algorithm for an STSG with a distribution U^X of subtrees having up to 3 lexical items. Note that the amount of discounted probability mass varies across submultisets. As the number i of words increases, it becomes harder and harder to find the same event in the distribution. Thus, as i increases, N_1 also increases. Since N_1 is the amount of counts discounted by the Good-Turing method, the amount of probability mass discounted from those higher-order submultisets increases as well.

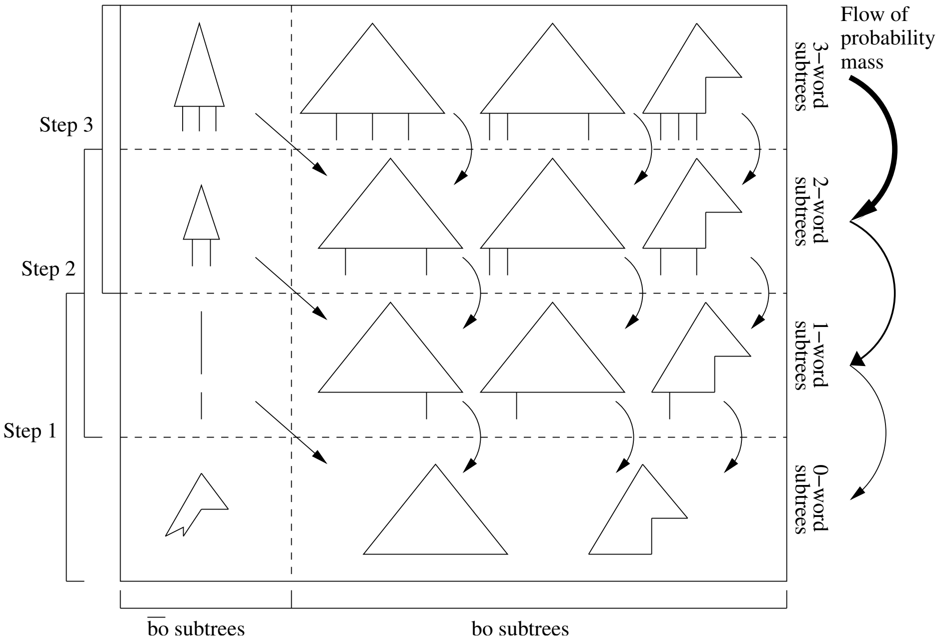

Figure 3.3: Schematic view of the back-off estimation algorithm. The dashed lines define partitions in the space of X-rooted subtrees. In this example, the maximal number of lexical items per subtree is 3. Note that the amount of probability mass discounted varies across the subspaces.

3.3 Previous work

The back-off model proposed in this thesis differs completely from the back-off models previously used in language modelling and DOP models.

In its n-gram formulation, probability mass is discounted from seen events and distributed to unseen events according to some lower-order distribution. This means that an unseen event would get probability zero if no probability mass were reserved for it. In the back-off model used here, the ‘unseen’ event – a previously unseen parse tree – can have its probability estimated by means of longer derivations. For all 2-word subtrees in the model, there are two corresponding 1-word subtrees which were generated from it, and these can generalize over the parse trees from which the model was extracted. So, in BO-DOP, the smoothing effect of avoiding zeroes is less important.

More important is the probability mass flow from n-word subtrees to (n-1)-word subtrees. In n-gram back-off models, smoothing was helping the model to be robust to unknown events, whereas here 'smoothing' is working as a correction factor, tying up events that, otherwise, would be regarded as disjoint.

In the following we compare BO-DOP with two previously defined DOP models (DOP3 and Tree-grams) that used smoothing in their implementations.

3.3.1 DOP3

Bod (1996) adapted Good-Turing smoothing technique to DOP, creating a model called DOP3. To estimate the number of unknown events, DOP3 followed three assumptions: i) no subtree was larger than depth 3; ii) all unlexicalized subtrees were known, implying that unknown subtrees were the ones having unknown lexical items; iii) the size of the vocabulary was known.

The number of unknown subtrees of a given distribution (e.g. NP-rooted distribution) was estimated by counting all possible combinations of lexical items to the preterminal nodes of the unlexicalized subtrees in the distribution. For NP-subtrees, 1.10260×10^9 possible distinct subtrees were estimated. From this number, 77,479 observed subtrees were subtracted, resulting in $N_0 = 1.10253 \times 10^9$. The overall number of counts reserved by the Good-Turing method is N_1 (see Chapter 2, Section 3.2). Thus, the count of each unknown event was obtained by dividing the total number of discounted counts by the estimated number of unknown events ($N_1/N_0 = 60,416/1.10253 \times 10^9 \approx 5.5 \times 10^{-5}$). Table 3.1 summarizes the results.

N_r	r	r^*
1.1×10^9	0	5.5×10^{-5}
60416	1	0.30
9057	2	1.37
4161	3	1.86
1944	4	1.99
773	5	3.74
482	6	4.37

Table 3.1: Adjusted frequencies for NP-rooted trees in Bod (1996).

DOP3 model was then obtained by extending the subtree space with the expected unknown subtrees and their associated GT counts[10]. This procedure differs from BO-DOP in some respects. First, BO-DOP does not extend the space of subtrees: the subtrees in DOP1 and in BO-DOP1 are identical[11].

[10]In practice, the space was extended with special subtrees that were able to treat one or more terminals as wildcards, allowing for all possible combinations.

[11]BO-DOP1 is DOP1 model with a back-off estimator. See Chapter 4, Section 3.

Second, the reserved mass was distributed uniformly (see Chapter 2, Section 4 for shortcomings of this approach). In DOP3 smoothing, probability mass is simply discounted and given to all possible unknown events. In our approach, probability mass is reallocated in a gradual and hierarchical way.

3.3.2 Tree-grams

Sima'an (2001a) presents a back-off procedure that increases the robustness of previous DOP models by making use of a Markov grammar (see Chapter 1, Subsection 1.5). In his model, called Tree-gram model, rules and corresponding probabilities are generated as needed. The generation of head-sister nodes for depth-1 subtrees follows a 1st-order Markov process. If it is not possible to estimate the probability of a rule with a 1st-order assumption, the model is backed off to a 0th-order model.

The Tree-gram model also allows backing off on node label tags. These tags can indicate, for instance, the 'role' of a subtree (e.g. a subtree containing a head-word has its role set to 'head'). If the probabilities defined on a distribution rooted at a tagged label cannot be properly estimated, the estimation is backed off to a less specific distribution, when more reliable statistics can be used. The probabilities of these subtrees can then be calculated as follows: if $P(t|X\text{-}Y) < k$, where Y is a tag associated with a root node X and k is small and fixed, then $P(t|X\text{-}Y) \approx \alpha(Y)P(t|X)$. The rule X-Y $\rightarrow$ X is added to the distribution and the normalization factor α is calculated using a Good-Turing-like function. The resulting back-off subtrees are included in the model together with the original ones.

The Tree-gram smoothing procedure differs from our approach in two aspects. First, Tree-grams back off on non-terminals and on label tags. Our model, on the other hand, backs off on lexical items. Second, the motivation underlying the Tree-gram model is to enlarge the space of derivations for DOP, increasing robustness; whereas in BO-DOP, the goal is to distribute probability mass across derivations that are back-offs of one another.

In the next chapter we compare the performance of DOP1 and Bon-DOP models with the results obtained from their back-off versions.

Chapter 4

Empirical results

In this chapter we present empirical results obtained with the Back-Off DOP estimator when associated with DOP1 and Bon-DOP models. The experiments were carried out in the OVIS corpus. In the following, we describe the corpus, define the evaluation metrics used to assess performance and compare results across different models.

4.1 Ovis corpus

OVIS is a Dutch language, speech-based dialogue system whose main goal is to provide railway time-table information to human users over ordinary telephone lines[1]. It is made up of different modules, including a Dialogue Manager, a Speech Recognizer, a Natural Language Processing (NLP) module and a Language Generation module. The NLP module is an interface between the Speech Recognizer and the Dialogue Manager. The output of the Speech Recognizer is processed by the NLP module, which extracts the semantic content of the user utterances. This information is taken by the Dialogue Manager, which updates its information state and decides on the next step (e.g. ask one more question, give information).

4.1.1 Compositional semantics

The OVIS corpus consists of over 10,000 syntactically and semantically annotated user utterances. These are answers given by users to the system's questions, such as "From where to where do you want to travel?" or "At what time do you want to travel from X to Y?". The answers are used to fill in a number of information slots that are typical of travel information, such as origin, destination, date and time. The semantic content of the user's utterances is used to

[1] OVIS stands for Openbaar Vervoer Informatie Systeem (Public Transport Information System). The pilot version is based on a German system developed by Philips in Aachen, Germany. The system is up and running and can be reached in the Netherlands at 0900-9292.

update the system's information state.

The semantic annotation is based on an *update language* defined to the OVIS Dialogue Manager by Veldhuizen van Zanten (1996). This language defines a hierarchy of slots and values. For the sentence IK WIL NIET VANDAAG MAAR MORGEN NAAR UTRECHT ("I don't want to travel today but tomorrow to Utrecht"), the update expression is given by

```
user.wants.(([#today];[!tomorrow]);destination.place.town.utrecht)
```

The symbol '#' means that the information between square brackets is incorrect, while '!' represents the correct information. Thus, the update language allows for encoding speech-act behaviors, such as denials. A special category ERROR enables the system to handle stutters and false starts, typical of spoken utterances. The update language is used to enrich OVIS trees with semantic information. The following annotation convention is applied: i) every *lexical* node is annotated with a slot and a value, representing the meaning of the lexical item; ii) every *non-lexical* node is enriched with a *formula* indicating how its meaning can be composed from the meaning of its daughter nodes. Figure 4.1 shows an example of an OVIS tree enriched with semantic information.

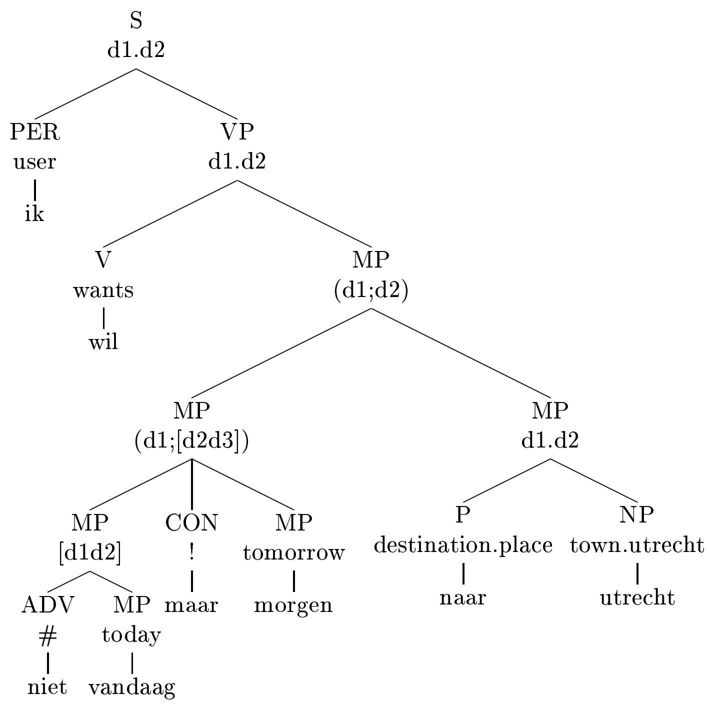

Figure 4.1: OVIS tree enriched with semantic information

The semantics of the OVIS corpus is compositional. Part-of-speech tags are enriched with semantic expressions (e.g. NP-town.utrecht) and the meaning of a non-terminal node is expressed in terms of the meaning of its daughters. The meaning of the whole sentence is then obtained in a bottom-up fashion by means of a few composition operations. If di represent a non-terminal's i-th daugther, then (d1;d2) is the concatenation of update expressions d1 and d2 and d1.d2 is the statement that slot d2 is more specific than slot d1.

Bonnema (1997) describes a method to turn semantic expressions into labels. This enables OVIS trees to be represented as regular phrase-structure parse trees. The semantic expressions are classified according to the type of their slots. Place expressions, for example, define one label category, while time expressions define another category. This process is able to retain unambiguously the semantics of the trees. In this work, we use the OVIS corpus annotated with this categorization method. A corresponding syntactic version of the corpus can be obtained by stripping the labelled nodes from their semantic tags.

4.1.2 OVIS in numbers

The corpus contains 10,049 parse trees. For the experiments reported below, the corpus was randomly split into two sets: i) a training set with 9,049 trees; ii) a test set with 1,000 trees. All experiments, unless stated otherwise, were carried out using the same train/test split.

Most trees in the OVIS corpus have less than 6 words. About 1/3 of the corpus is made up of 1-word trees. Figure 4.2 shows how trees are distributed with respect to the number of words in their frontiers.

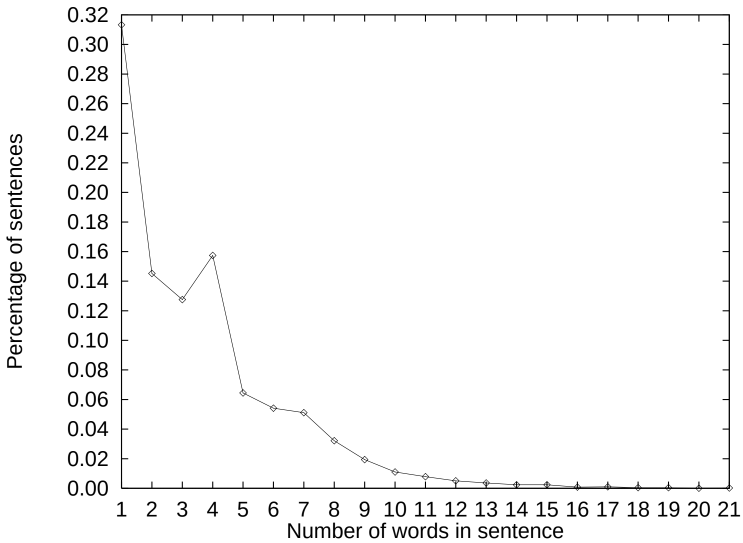

Figure 4.2: Percentage of sentences as a function of their number of words.

The OVIS corpus is relatively homogeneous. Sentences with 4 words are more common than sentences with 2 and 3 words, since there are many constructions of the form VAN X NAAR Y ("From X to Y"). The average number of words per sentence is 3.47. This is contrast to the Wall Street Journal corpus, whose trees have an average sentence length of 23 words and belong to texts of various domains, resulting in much less homogeneity.

To obtain more meaningful results, we removed all 1-word trees from the test set. Statistics were collected from the complete corpus, but only sentences with more than 1 word were tested. Without 1-word sentences, the average sentence length relative to the whole corpus is 4.6.

Annotation	Grammar symbols		
	Non-terminals	Preterminals	Terminals
Syn+Sem	548	248	965
Syn	150	127	965

Table 4.1: OVIS treebank in numbers

Table 4.1 shows the numbers of terminals (words), preterminals (part-of-speech tags) and nonterminals in the corpus. Preterminals are a special case of non-terminals and are included in their counts. The semantic annotated corpus has $(548 - 248) = 300$ non-preterminal non-terminals, whereas the syntactic corpus has only $(150 - 127) = 23$. This means that the semantic corpus defines over 10 times as many distributions as the syntactic corpus, yielding much more fine-grained estimates. This greatly influences the relative performance of the corresponding models.

4.2 Metrics and parameters

We describe here the criteria used to evaluate BO-DOP performance.

4.2.1 Accuracy measures

A model's generative power is measured by the coverage of its associated grammar. Note that DOP1, Bon-DOP and associated BO-DOP models all have the same coverage, since their subtree spaces contain the same subtree types.

- **Coverage:** the percentage of sentences in the test set for which a parse was found. $\texttt{Coverage} = \frac{\texttt{\#proposed parses}}{\texttt{\#parses in test set}} \times 100$

The most important accuracy measure used in this thesis is the exact match, since the parser we used was designed to maximize this metric (Sima'an, 1999).

- **Exact Match:** the percentage of parses assigned by the parser which exactly match their test set counterparts. This is the most strict measure used. $\texttt{Ex.Match} = \frac{\texttt{\#correct proposed parses}}{\texttt{\#parses in test set - \#failed parses}} \times 100$

The exact match criterion is straightforward and gives a measure of the parser's accuracy. However, it misses information about constituents that the parser may hypothesize correctly but that do not yield a complete match. To overcome that, Black et al. (1991) proposed less strict accuracy measures. These are bracketing accuracy measures (Parseval) that are now standard in the NLP community. Three Parseval measures were used in this work:

- **Labelled Bracketing Recall:** the ratio between the number of correct labels assigned by the parser and the total number of labels in the test parse. This measures how detailed is the bracketing found by the parser.
  Labelled Recall (LR) $= \frac{\texttt{\#correct labels in proposed parse}}{\texttt{\#labels in treebank parse}} \times 100$

- **Labelled Bracketing Precision:** the ratio between the number of correct hypothesized labels and the total number of labels assigned by the parser. This measures how accurate is the bracketing found by the parser.
  Labelled Precision (LP) $= \frac{\texttt{\#correct labels in proposed parse}}{\texttt{\#labels in proposed parse}} \times 100$

- **F-score:** the harmonic average of recall and precision results.
  $\texttt{F-score} = \frac{2 \times \texttt{LP} \times \texttt{LR}}{\texttt{LP} + \texttt{LR}}$

Exact match, recall and precision all refer to parses that were actually *found* by the disambiguator. This means that failed parses were not counted. Hence

$$|\texttt{test set}| = \texttt{\#parses in test set - \#failed parses}$$

We used a non-standard measure to verify the effects of the back-off estimator on the number of substitutions.

- **N.Subst.:** the total number of substitutions in the MPD parses returned by the disambiguator;

- **Av.Subst.:** the average number of substitutions per sentence in the test set.
  $\texttt{Av.Subst.} = \frac{\texttt{N.Subst.}}{|\texttt{test set}|}$

Relative performance among models can be calculated by means of *error reduction ratios.* They measure how much one model improved on the other model relative to the latter's global error rate:

- **Error Reduction:** the ratio between the percent points improvement of model 1 over model 2 normalized by the global error of model 2
  $\texttt{Error red.} = \frac{\texttt{\%accuracy of model 1 - \%accuracy of model 2}}{\texttt{100\% - \%accuracy of model 2}}$

4.2.2 Disambiguator parameters

Sima'an (1999) suggested four constraints on the shape of subtrees in order to restrict the subtree space to a manageable size. For the experiments reported in this thesis, we made use of the following constraints:

- **l :** maximum number of lexical items;
- **n :** maximum number of substitution sites;
- **d :** maximum subtree depth;
- **L :** maximum number of consecutive lexical items

A distribution extracted with parameters l2n4, for example, have subtrees with at most 2 words and at most 4 substitution sites.

Another experimental parameter is the maximization entity used to select the most likely parse. Two methods were used here[2]: i) Most Probable Derivation (**MPD**); ii) Most Probable Parse (**MPP**). Both are described in Chapter 1. For experiments using MPP, we also define the Monte Carlo sample size. A sample of 5,000 derivations, for instance, means that the probability of the proposed parse was found by summing up the estimates of 5,000 derivations drawn randomly from the derivation space.

4.3 Results

For every DOP model, a corresponding BO-DOP model can be derived by applying the procedure described in Chapter 3. We tested the back-off model on two DOP models: DOP1 and Bon-DOP. The results reported here are referred to according to the following convention:

- **DOP1:** the DOP1 model, defined in Chapter 1;
- **Bon-DOP:** the Bon-DOP model, defined in Chapter 1;
- **BO-DOP1:** the DOP model obtained by applying the back-off estimation procedure to the DOP1 estimator;
- **BO-Bon-DOP:** the DOP model obtained by applying the back-off estimation procedure to the Bon-DOP estimator

We carried out experiments on both semantic and syntactically annotated OVIS corpora. One important issue is the handling of unknown input words. We did *not* use any smoothing procedure to tackle unknown words. All results reported here come from parsers trained on the complete closed vocabulary set. This differs from the approach used by Sima'an (1999), who implemented an Add-One smoothing procedure to assign non-zero probabilities to previously unseen words. Moreover, the results reported here are not comparable to some previously published results, since the corpus has been recently modified to correct for inconsistencies (Sima'an, personal communication). The complete set of experimental results can be found on **Appendix B**.

[2] A third possible maximization entity is the *Maximum Constituents Parse*, which corresponds to the parse that is most likely to have the largest number of correct constituents. Goodman (1998) developed an algorithm that maximizes this entity.

4.3.1 Semantic corpus

We first report results obtained on the semantic corpus.

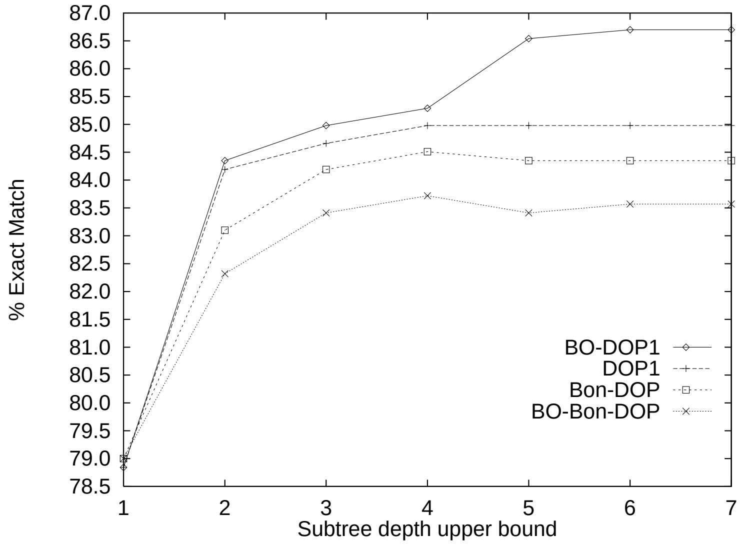

Figure 4.3: Exact match as a function of depth upper bound (l2n4,MPD).

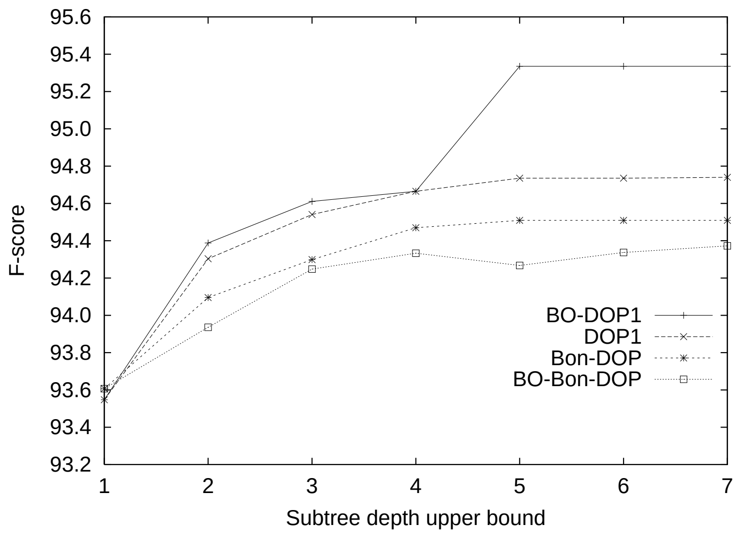

Figure 4.4: Bracketing accuracy as a function of depth upper bound (l2n4,MPD).

Varying subtree depth

Figure 4.3 shows exact match results as a function of subtree depth upper bound. The subtrees were restricted to at most 2 words and 4 substitution nodes. This yields small subtree spaces: for depth 7, this corresponds to 172,050 subtrees.

For all depth upper bounds, BO-DOP1 achieved the best results followed by DOP1, Bon-DOP and BO-Bon-DOP. BO-DOP1 improved on DOP1 by 1.72 percent points at depth 6, an increase of 2.02%. This corresponds to an error reduction of 11.4%. When compared to Bon-DOP, error reduction rose to 15%.

F-score results follow the same pattern. At depth 6, BO-DOP1 reached 95.33%; DOP1, 94.73%; Bon-DOP, 94.5% and BO-Bon-DOP, 94.33%. Error reduction of BO-DOP1 with respect to DOP1 reached 11.3% and with respect to Bon-DOP, 15.7%.

For the rest of this chapter, we empirically test some properties of the back-off models and report more results obtained with different parameter settings.

Discounted probability mass

As pointed out in Chapter 3, the smoothing procedure has the property that i-word subtree subdistributions tend to contribute more to the discounted probability mass N_1/N than (i-1)-word subdistributions. This follows from the fact that larger subtrees are more likely to appear just once than smaller subtrees (subtrees that occur just once are called *singletons*).

For higher depth upper bounds, the number of singletons tends to increase, since more large subtrees are allowed in the distribution. More singletons means higher N_1 values and, therefore, higher discounting rates.

Figure 4.5 shows discounted probability mass as a function of subtree depth upper bound. As the upper bound increases, larger 2-word subtrees start to populate the distribution. Since these are less likely to occur again, they will increase the relative contribution of 2-word subtrees to N_1.

Note that for small depths, 1-word subtrees contribute more to N_1 than 2-word subtrees. This is a consequence of the small number of 2-word subtrees in the distribution. Lexicalized unary subtrees end up taking a larger share of the discounted mass. Increasing the depth upper bound has the effect of enlarging the distribution with many large and unlikely 2-word subtrees, diluting the relative contribution of unary subtrees.

Figure 4.6 presents similar results with parameters l3n4. Observe that the discount rates were shifted upwards. By allowing larger subtrees in the distribution (3-word subtrees), the number of singletons tends to increase, expanding the discounted mass. Like the previous case, 1-word and 2-word subtrees tend to contribute more to N_1 for small depth upper bounds.

An important feature of the back-off models discussed here is that the transfer of probability mass differs in magnitude across models: in general, BO-DOP1 transfers much more probability mass among subtrees than BO-Bon-DOP does. This is a consequence of Bon-DOP's estimation procedure. Small subtrees tend to have much higher probabilities than big ones according to Bon-DOP's esti-

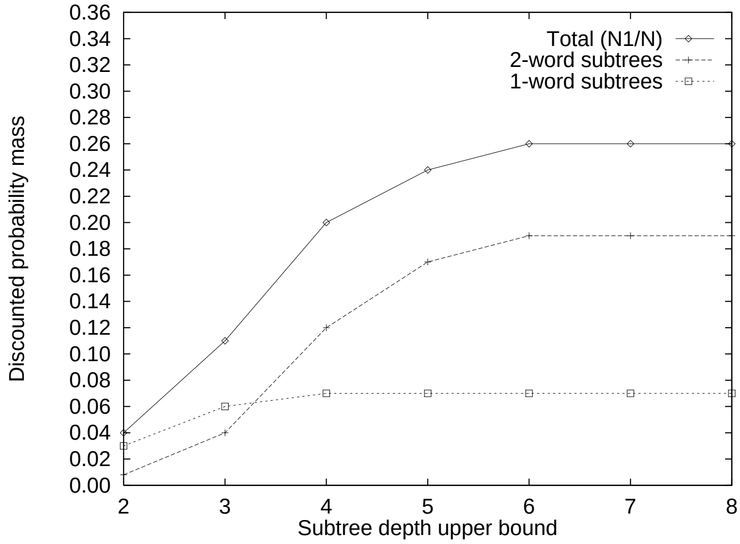

Figure 4.5: Probability mass discounted from n-word subtrees as a function of subtree depth upper bound (l2n4).

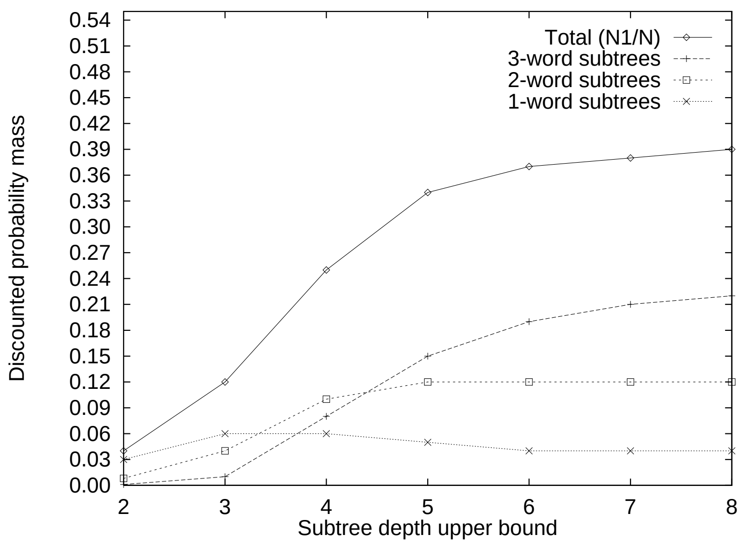

Figure 4.6: Probability mass discounted from n-word subtrees as a function of subtree depth upper bound (l3n4).

mator, since the latter will have their counts reduced by a scaling factor $2^{-N(t)}$ that is exponential in subtree size. As a result, the denominator of the correction factor $\alpha(X, i)$ for BO-Bon-DOP models tends to be higher than for

BO-DOP1, yielding smaller $\alpha(X, i)$ values. Thus, lower-order subdistributions in BO-Bon-DOP tend to receive less probability mass than in BO-DOP1.

In addition, because several counts in BO-Bon-DOP will be fractionary, many distributions will end up not being discounted at all. So, not only the discounting rate, but also the number of discounted distributions is smaller in BO-Bon-DOP. Consequently, BO-Bon-DOP estimates are not very different from Bon-DOP estimates. This slight change, however, is enough to drop BO-Bon-DOP's performance. We will return to this point in Chapter 5.

Number of substitutions

Discounting probability from subtree distributions has a direct effect in the size of derivations. This can be verified by looking at the average number of substitutions in the MPD returned by the parser. The results are summarized in Table 1 of Appendix B. Back-off discounting can be detected by observing that the number of substitutions in a BO-DOP tends to be larger than the substitutions in the associated DOP model. This is valid for all depths. For increasing depth upper bounds, notice that the average number of substitutions decreases. That is because more large subtrees start to participate in the MPD, resulting in shorter derivations.

Most Probable Parse

The next set of experiments used distributions obtained with parameters l7n2L3. These settings allow testing the models quickly, since they generate small subtree spaces. For depth 7, there are 133,308 subtrees. Another advantage is that we can observe the effects of longer back-off cascades, since the models could, in theory, back-off from 7-word subtrees to 0-word ones. In practice, however, most cascades will start from 4-word subtrees. This is a consequence of OVIS' sentence length distribution, whose mean is less than 5 words/sentence.

Figures 4.7 and 4.8 show exact match results as a function of depth upper bound and maximization entity (see also Tables 2 and 3 in Appendix B). Note that they follow the same pattern observed in l2n4 results. Unlike those, however, the results for l7n2L3 achieved maximum accuracy at depth 3, not 6.

Note that MPP has the effect of increasing performance for most models when compared to MPD. This shift is stronger for DOP1 and BO-Bon-DOP. One possible explanation is that MPP allows for recovering part of the joint dependencies lost by the independence assumption underlying MPD. Moreover, MPP can return the correct parse when the assumption that the probability of a parse is concentrated in a single derivation does not hold.

Different splits

To test whether the results above are due to some random property of the train/test split, we took a 'cross-section' of the results for l7n2L3 at depth 3 and changed the splits. The number of trees was kept constant: 9,049 for training;

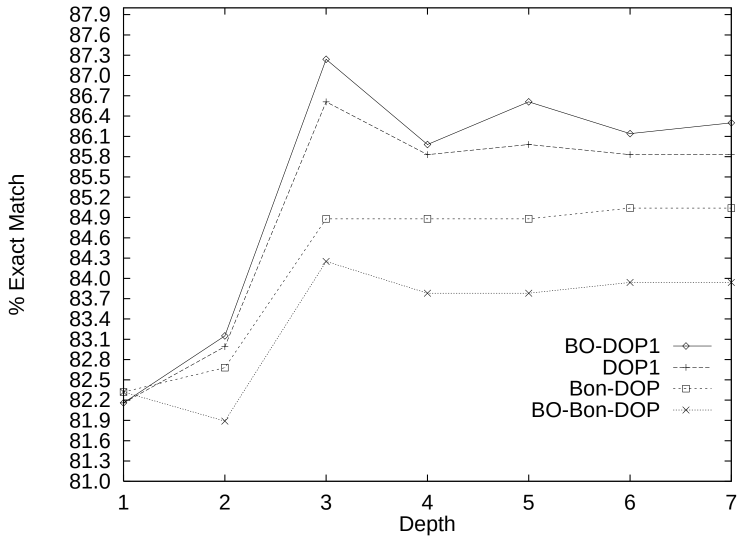

Figure 4.7: Exact match as a function of depth upper bound (l7n2L3,MPD).

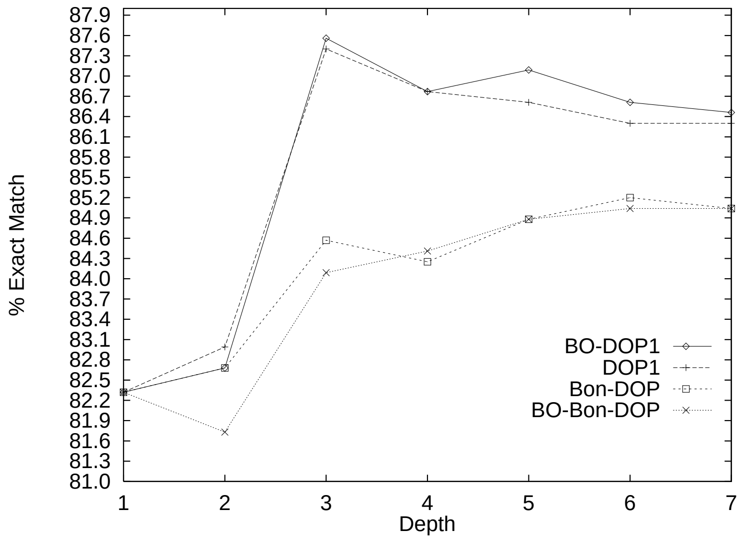

Figure 4.8: Exact match as a function of depth upper bound (l7n2L3,MPP).

1,000 for testing. Depth 3 was chosen because most models achieved their best results with this setting, which might indicate some random fluctuation. Four experiments were carried out. Experiment 1 refers to the split used throughout this work. Experiments 2, 3 and 4 refer to different splits obtained through random drawings from the OVIS semantic corpus. Note that different splits yield distinct models with distinct generative powers (see Coverage in Table 4, Appendix B). The results with different splits are shown in Figure 4.9. The pattern previously observed, with BO-DOP1 achieving the best results and BO-Bon-DOP the worst, is conserved across the splits.

In this small sample of 4 experiments, BO-DOP1 relative improvement over DOP1, although persistent, is not statistically significant. BO-DOP1's performance mean of 87.10% is not significantly different from DOP1's mean of 86.54% with 95% confidence according to the t-test (interval: $\pm 0.3097\%$)[3].

Experiment 3 was the only one that reached statistical significance. It is important to emphasize that the wide range of the confidence interval is due to the small sample size. Definitive conclusions can only be drawn once a larger number of experiments is carried out. Moreover, these results refer to a single 'cut' in the parameter space and it is possible that different parameter combinations will result in more significant differences. In any case, the relative ranking between BO-DOP1 and DOP1 is not contradicted by any of these results.

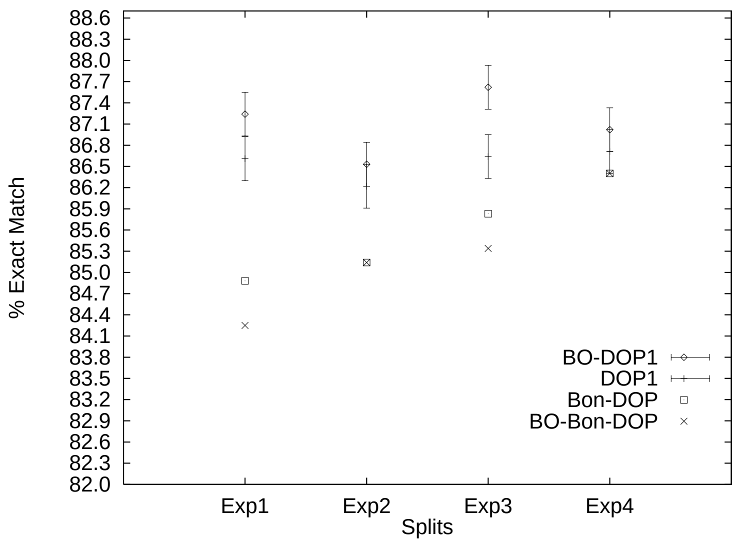

Figure 4.9: Exact match for fixed depth 3 as a function of training/test set splits (l7n2d3L3,MPD)

Large distributions

One could argue that the distributions used so far are favoring DOP1, since they are very small and counteract DOP1 bias towards large subtrees. To assess the behavior of these models under larger subtree spaces, we generated subtree distributions with parameters l5n5. For depth 7, this yields a space containing

[3]The t-test is a hypothesis test for normal distributions. Tree distributions are *not* normal, due to their Zipfian nature. Yet t-test may give useful information about the sample means.

993,483 subtrees, a 7-fold increase over the corresponding distribution with parameters l7n2L3. Figures 4.10 and 4.11 show the results for depths 4, 5 and 6. We did not run experiments with higher depths due to time limitations.

The results for MPD show a slight fall in DOP1's performance when compared to corresponding results obtained with parameters l7n2L3 and l3n4 (see Table 7 in Appendix B) and a short rise when compared to l2n4 results. The other models followed suit. The ranking pattern remained unchanged.

As to MPP, l5n5 results show a general increase in performance for most models when compared to MPD results. As with l7n2L3 distributions, MPP has the effect of reducing the distance between DOP1 and BO-DOP1 and between BO-Bon-DOP and Bon-DOP.

These results, however, do not reflect DOP1 biases. To seriously evaluate that, we need much larger subtree spaces. OVIS sentences and trees are too small, reducing considerably the exponential explosion expected when using less restrictive parameters. Moreover, the constraints imposed here are still too restrictive. The example illustrating DOP1 bias in Chapter 1 (Figure 1.6), for instance, only works if no restrictions are imposed. By constraining the maximal number of substitution sites, DOP1 already manages to recover the correct parse in the example, operating as if it had no bias.

4.3.2 Syntactic corpus

We also tested the models on the syntactic version of the OVIS corpus. Figure 4.12 compares exact match performance of models trained on syntactically and semantically annotated corpora. Parameters were set to l3n4 and the maximization entity used was MPD. Results are presented together in the same figure to facilitate the comparison of accuracy across annotation schemes.

Figure 4.12 shows that models trained on syntactic OVIS perform consistently worse than models trained on semantic OVIS. For all depth upper bounds, semantics outperforms syntax.

One possible reason is that the distributions in semantic OVIS are more fine-grained than the ones in its syntactic counterpart (300 distributions versus 27), enabling them to guide the disambiguator more precisely during parsing. Distributions on the syntactically annotated corpus, even though less sparse, are made up only of 'average' estimates. In the trade-off between more data per distribution and more distributions per model, the latter won.

BO-DOP1 relative advantage over DOP1 was reduced in the syntactic corpus. One explanation is that there is more discounting in syntactic distributions than in semantic ones. This follows from the fact that syntactic distributions are less sparse. Thus, most syntactic distributions will have their counts discounted. On the other hand, only a small fraction of the semantic distributions are discounted, since most of them are made up of very few subtrees, which prevents Simple Good-Turing from being used.

The increase in discounting rate among syntactic models, causes an increase in the number of longer derivations returned by MPD. Since the probabilities underlying these derivations are more blunt compared to semantic probabilities, the parser tends to make more 'guesses' and, eventually, more *wrong* guesses.

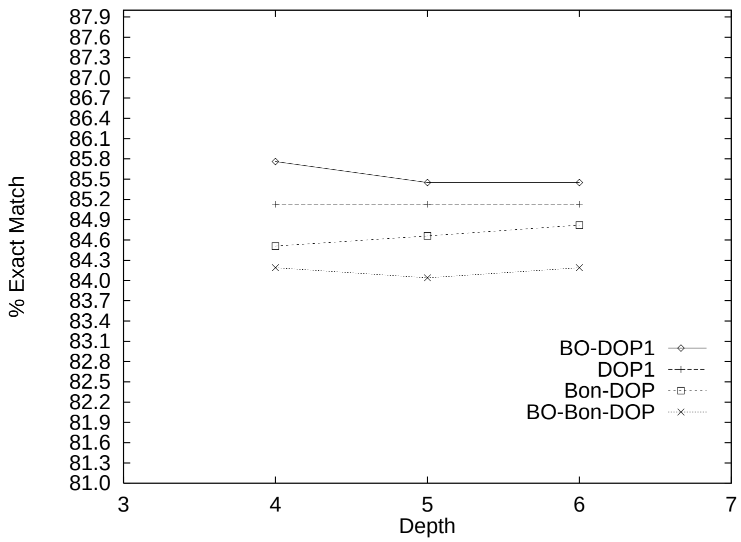

Figure 4.10: Exact match as a function of depth upper bound (l5n5,MPD).

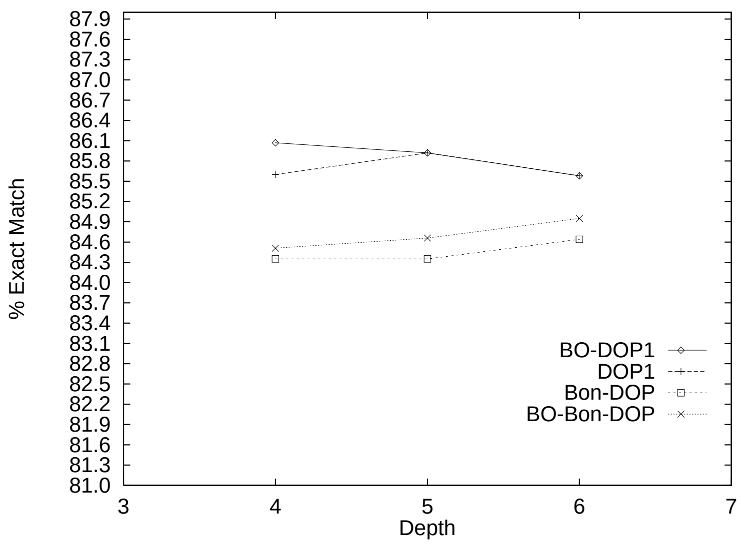

Figure 4.11: Exact match as a function of depth upper bound (l5n5,MPP).

Moreover, because OVIS is so homogeneous, large subtrees encode disproportionately more correct joint dependencies than small subtrees (in bigger corpora,

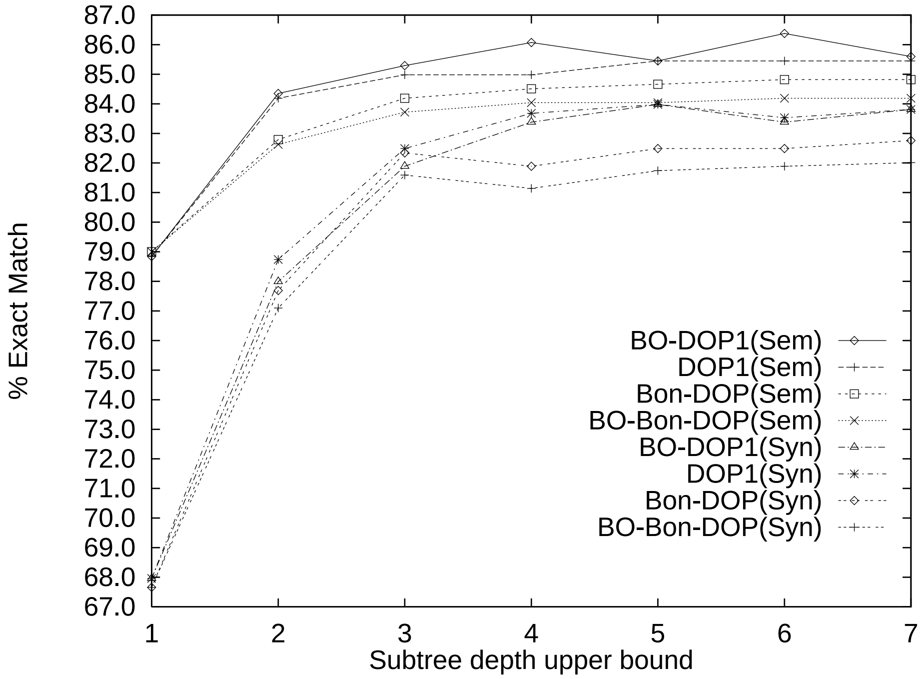

Figure 4.12: Exact match as a function of subtree depth upper bound for both semantic and syntactic corpora (l3n4,MPD).

large subtree structures are so unfrequent, that the information they encode loses relevance; in OVIS, on the other hand, there are several recurrent large structures). Because syntactic OVIS distributions are considerably discounted, longer derivations with shorter subtrees gain prominence, reducing the relative importance of large subtrees and decreasing performance.

One indication that more combinations are allowed in syntactic OVIS comes from the fact that it can generate more parses than semantic OVIS (compare Coverage in Tables 7 and 8 of Appendix B). This enhanced generative power is translated into more subtree combinations. As mentioned earlier, because of DOP1 and Bon-DOP's weak lexicalization problem, the chances of making wrong combinations tend to increase.

Note that the performance pattern observed in all previous experiments in the semantic corpus is mostly conserved in the syntactic corpus. In addition to that, DOP1 outperformed Bon-DOP throughout the experiments. This suggests that, if one has to choose among two estimators – one favoring large subtrees, the other favoring short ones – one should pick the first.

In the next chapter, we offer a tentative account for the results reported here in light of the properties of BO-DOP and the biases of DOP1 and Bon-DOP.

Chapter 5

Conclusions and Discussion

We started this thesis aiming at tackling two shortcomings of current DOP models: i) the biased estimation procedure in DOP1; ii) the assumption that derivations are equally likely in Bon-DOP. We proposed to attack these problems by means of a smoothing-like procedure: taking probability mass from 'seen' events in a distribution and reallocating it appropriately to 'unseen' events.

In subtree distributions, 'seen' and 'unseen' events were translated in terms of derivations. The smallest possible derivation – the parse tree itself – is given maximal priority. That is our 'seen' event. In case such a derivation is not available, the model would back off to the next shortest derivation(s). That is our 'unseen' event. The process then follows iteratively until the last derivations[1], which are the largest ones, are found.

Unlike standard back-off, however, the method described here does not aim at smoothing a subtree distribution. It does not make the model more robust to previously unseen events. So, it is not the case that a back-off event is only taken into consideration if the original event is not available, since back-off derivations are generated *along with* the higher-order ones, and the most probable derivation is the one that wins the disambiguation race. The only role played by back-off in our model is to provide a principled way of creating an ordering in the derivation space. This ordering is based on the assumption that subtree distributions and the corresponding derivations extracted from treebank parse trees are related to each other. This is in contrast with current DOP probability estimators, that regard subtree events as unrelated. To tie up derivations, we resorted to back-off. Thus, what back-off does in practice is simply to change the relative frequencies of derivations by moving probability mass across the derivation space.

The back-off technique we implemented is related to points i) and ii) in the following way: discounting probability from 'seen' events amounts to taking probability from larger subtrees, which alleviates DOP1's bias towards large subtrees; giving probability mass to 'unseen' events, i.e. the next derivations in

[1]For a given parse tree, there maybe be several last derivations depending on the route taken in the back-off hierarchy. Figure 3.2 in Chapter 3, for instance, shows that there are two possible last derivations. Note, however, that there is only one *largest* derivation per tree.

the hierarchy, has the effect of changing the relative likelihood of derivations.

The procedure was tested on DOP1 and Bon-DOP models and yielded consistently better results when applied to DOP1 and consistently worse results when applied to Bon-DOP. To account for that, we hypothesize the following:

1. BO-DOP1 improved on DOP1 because the probability mass it discounted is big enough to reduce DOP1's bias towards large subtrees but small enough to prevent the model from favoring small subtrees;

2. DOP1 bias towards large subtrees is translated into better performance than Bon-DOP bias towards small subtrees because big structures are able to capture more linguistic joint dependencies;

3. BO-Bon-DOP performs worse than Bon-DOP and all other models because it aggravates even further Bon-DOP's bias

Support for hypotheses 1 and 3 comes from the way probability mass is distributed in each model. Hypothesis 2 is supported by previous results showing that an absolute bias towards large subtrees might even increase DOP model's performance. In the remaining of this chapter we present evidence favoring those hypotheses, speculate on why DOP1 performance should possibly *not* be impaired by increasing subtree spaces and discuss further research directions.

5.1 Comparing distributions

DOP1 and Bon-DOP models favor large and small subtrees in opposite ways. By applying a back-off estimation procedure to these models, the underlying distributions change. But the changes the procedure cause have also opposite effects: whereas for DOP1 back-off reduces bias, for Bon-DOP it increases bias even more. If $c(t)$ represents the frequency counts of subtree t in a given distribution, then, for each model, the frequency estimate of t is given by

- **DOP1:** $c(t)$
- **BO-DOP1:** $d(c(t)) \cdot c(t) = c(t)^*$
- **Bon-DOP:** $2^{-N(t)} \cdot c(t)$
- **BO-Bon-DOP:** $d(2^{-N(t)} \cdot c(t)) \cdot (2^{-N(t)} \cdot c(t)) = (2^{-N(t)} \cdot c(t))^*$

where $d(r) = \frac{r^*}{r}$ and $r = c(t)$ (see *discount ratio* in Section 4.2, Chapter 2). Note that BO-DOP1's frequency estimation is based on subtree counts obtained directly from the model's extraction procedure. Note further that the discount rate is *linear*, since Good-Turing discounting is linear.

BO-Bon-DOP's estimation, on the other hand, is applied on top of Bon-DOP model, whose subtree counts have already been discounted by a factor that is *exponential* on subtree size. The exponential nature of this scaling factor may be the underlying reason why Bon-DOP appears to be biased towards smaller subtrees. By discounting Bon-DOP's counts, BO-Bon-DOP underestimates even further large subtree frequencies.

The underestimation is even more dramatic if one notices that the actual discounting experienced by Bon-DOP is very small, given its fractionary counts and data sparseness (see Section 3.1, Chapter 4).

As argued in Chapter 4, giving too much probability to small subtrees causes a decrease in performance because the parser starts to return longer MPDs. Longer MPDs are made up of several combined fragments, and since these combinations are only loosely constrained, they might not respect important linguistic restrictions. This is specially noticeable in syntactic OVIS, where there is still more freedom for spurious combinations.

Figure 5.1 compares schematically the flow of probability mass across models for each estimation procedure. Note that in the case of DOP1, the transfer is bottom-up, since large subtrees extracted from a treebank end up 'stealing' probabilities from small subtrees due to the exponential nature of the extraction process. In contrast, Bon-DOP transfer is top-down, since large subtree estimates are downgraded by the scaling factor $2^{-N(t)}$. Both back-off models present a more flexible probability transfer, giving less and less probability to subtrees down in the distribution. But while BO-DOP1 does so by taking from large overestimated subtrees in DOP1, BO-Bon-DOP enhances even more Bon-DOP underestimation. This scheme is consistent with hypotheses 1 and 3.

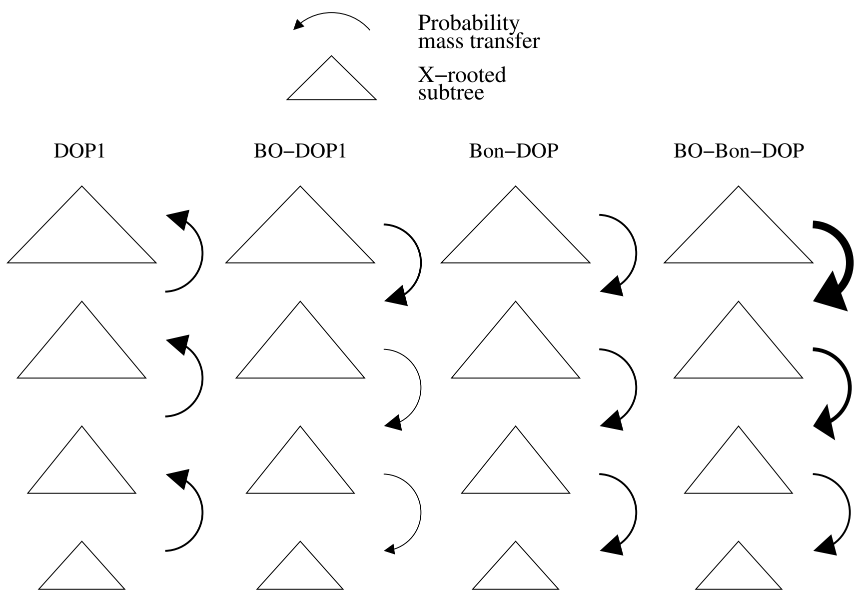

Figure 5.1: Schematic comparison of probability mass transfer for different DOP models within an X-rooted subtree distribution.

5.2 Shortest derivation

Bod (2000) reports experiments with a 'non-probabilistic' DOP model on OVIS and WSJ corpora. By 'non-probabilistic', it is meant a particular DOP model whose definition follows the definitions of DOP1 except for the estimation procedure: all subtrees in the model are given the same probability estimate p. This amounts to giving an absolute ordering to derivations: the most likely derivation is *always* the shortest one, since it involves less multiplications of p factors ($0 < p < 1$). When there is more than one shortest derivation, the model looks up a table that ranks subtrees by frequency, so that it can disambiguate and return a single shortest derivation. The 'non-probabilistic' model, therefore, takes the bias towards large subtrees to its extreme. The results on the OVIS corpus show that this model outperforms DOP1 for depth upper bounds[2] greater than 3. These results are in accordance with hypothesis 2.

Given the good performance of 'non-probabilistic' DOP, one might ask whether it is worth pursuing more sophisticated estimators when a simple heuristics can accomplish better results. The problem is that the OVIS corpus is highly homogeneous. So, large structures obtained from the training set will very much likely show up again in the test set. Thus, a model that gives total precedence to large structures will very likely guess it right during parsing. This would not be the case in wide-coverage corpora, like the Wall Street Journal.

In fact, the results reported for WSJ show that 'non-probabilistic' DOP performs *poorer* than DOP1 for most depth upper bounds. When taken to a large corpus, non-probabilistic heuristic models seem to lose ground for probabilistic ones. This is one important motivation to keep looking for better estimators.

The results we reported in this work need to be validated in a large corpus, such as the WSJ, in order to verify whether the improvements achieved here are real or simply an artefact. The fact that BO-DOP1 achieved consistently, even though slightly, better results than DOP1 on the OVIS corpus – whose structure and size greatly favor DOP1 – is already a good indication that there might be something correct about the model.

5.3 Local parsing decisions

One remaining question is whether DOP1's performance should indeed drop with increasing subtree spaces. Because of its bias towards large subtrees, one would expect that increasing spaces would result in worse performances. We could not conclude that from the experiments carried out in this thesis, since the spaces studied were still too small. Nevertheless, it might be the case that performance was not impaired because it should not be impaired anyway.

More speculatively, it is possible that most decisions concerning parse disambiguation are made by comparing subtrees of similar sizes. In this case, neither DOP1 nor Bon-DOP would suffer from increasing fragment space size, since the relations would be local and not global. The derivations competing for the

[2] The other parameters (l,n,L) were not specified in (Bod, 2000).

MPD would be of a similar order of magnitude and therefore it does not matter whether the grammar is huge or tiny, 'stealing' probability is not that important as long as it does not interfere too much with subtrees of similar sizes.

Consider the ambiguous sentence THE MAN SOLD THE DOG BISCUITS. The ambiguity lies in choosing between VP $\rightarrow$ V NP (sold [dog biscuits]) and VP $\rightarrow$ V NP NP (sold [biscuits] [to the dog]). The VP-rooted subtrees competing for the correct parse would probably be of about the same sizes. That is because if a large VP subtree is participating in a derivation d_1^i of a given parse (τ_1), it is very hard that a small VP subtree involved in a derivation d_2^j of a different parse (τ_2) wins the race, since it has to win several multiplications of p factors ($0 \leq p \leq 1$). Long VP-rooted subtrees yielding τ_2 would be a much stronger competitor, since the number of multiplications it would have to face are of the same magnitude as with d_1^i . So, in this view, what counts in disambiguating sentences is the relative frequency of structures with similar sizes.

Decisions regarding disambiguation would then be local, not global, in the sense that what makes a difference in disambiguation would be the differences in probability between subtrees in the same *size partition* of a distribution and not the probabilities between subtrees in different partitions. Long derivations would compete with long derivations; short with short. Being in different size partitions would imply comparing probabilities estimates of subtrees with very different sizes. In this case, the model's biases would start to play a role. In contrast, the bias effect would be considerably reduced if subtrees of similar sizes are the ones competing for the MPD. This would hint at why DOP1's performance does not go down with increasing subtree distribution size.

This view is consistent with Bon-DOP's poor performance with respect to DOP1. For Bon-DOP, even subtrees of similar sizes have large differences in probabilities. A subtree with 4 non-root nodes gets shrunk by a factor 2^{-4}; a subtree with 6 such nodes, shrinks by 2^{-6}; this already changes the relative frequencies by a factor of 4. If the effect of relative frequency is added (presumably more 4-node subtrees than 6-node ones in the distribution), then what may happen is that an alternative longer – and possibly wrong – MPD is chosen.

5.4 Future Directions

The model implemented in this thesis is a simplification of the back-off approach, since we took only lexical back-offs into account. Implementations able to cope with structural back-offs would give a much better view of the strengths and limitations of this approach. Another simplification was the assumption of closed vocabulary. No unknown words were allowed. This is certainly a major weakness of the current implementation. Nonetheless, it can be easily corrected by adding rules of the form POS-tag $\rightarrow$ UNKNOWN to the subtree space. The word emit probabilities could then be estimated from statistics collected from word-endings, hyphenation and capitalization (Weischedel et al., 1993).

On the theoretical side, it is important to assess the properties of the back-off estimator, including bias and consistency. This could shed light into new strategies to improve DOP models' parameter estimation.

BO-DOP1 could be extended by adding a real smoothing procedure to the model in order to increase its robustness. Following Lexicalized Tree-Adjoining Grammars (LTAGs), lexicalized subtrees could be grouped into families of subtrees sharing similar linguistic properties, such as subcategorization frames or passive structures (Chen and Vijay-Shanker, 2000). The intuition is that if a word seen in the training corpus is an anchor of a subtree belonging to a given family, then it is likely that this word would also anchor other subtrees in the same family. Thus, a DOP model extracted from a treebank could be extended by allowing lexicalized subtrees in the same family to swap their anchors. The probabilities of the new subtrees could be estimated by standard Good-Turing techniques. The new subtrees could be integrated in the model along with the original ones. A back-off estimation procedure could then be applied on top of the smoothed model. One possible issue is how to extend BO-DOP discounting procedure in order to cope with fractionary counts. Another issue would be how to identify and group lexicalized subtrees into linguistic families.

Despite the simplifying assumptions underlying the present work, the model obtained yielded promising results, opening the possibility to further tests in different DOP models, such as LFG-DOP, TIG-DOP or Tree-grams.

Tree-grams (Sima'an, 2001a), in particular, could represent a suitable testbed for BO-DOP. We blamed the poorer performance of BO-DOP1 in syntactic OVIS on two factors: i) less fine-grained distributions, when compared to semantic OVIS; ii) more discounting, leading to more wrong combinations due to DOP1's weak lexicalization. In other words, we blamed the poor results on DOP1's blindness about what is going on down in the parse tree during disambiguation. We believe head-lexicalized models, such as Tree-grams, could add to BO-DOP1 by percolating head information up in the parse tree, avoiding some of those spurious combinations. Conversely, the Tree-gram model could profit from BO-DOP, given that its estimation procedure suffers from the same biases as DOP1. Thus, we believe a Back-Off Tree-gram model could improve on BO-DOP1 much in the same way as Tree-grams improved on DOP1.

An alternative Back-Off DOP model

The occurrence of a subtree t in a subtree distribution can be seen as the joint occurrence of two subtrees t' and t'' such that $t' \circ t'' = t$. By the chain rule, we have $P(t|X) = P(t', t''|X) = P(t'|X)\, P(t''|X, t')$. The back-off smoothing formula for this distribution is given by

$$(5.1) \qquad P_{bo}(t|X) = \begin{cases} P'_{\mathrm{GT}}(t|X) & \text{if } c(t) > 0, \\ \alpha(X, t')\, P(t'|X)\, P(t''|Y) & \text{otherwise} \end{cases}$$

where P'_{GT} is a Good-Turing-type function and Y is a node such that: i) Y is a substitution node in t'; ii) Y is the root node of t''; iii) $t' \circ t'' = t$. In other words, if subtree t is seen, its probability is estimated through Good-Turing; if it was not seen, its probability is estimated by multiplying the probabilities of two smaller subtrees plus a correction factor that depends on the previous history. This is the approach taken by back-off techniques in language modelling. However, rather than backing off from X, t' to t', as language models do, we give here a much longer leap, backing off from X, t', Y to Y. Thus, we need to find a way of recovering lost contextual information.

If we assume that all subtrees t' are known, then smoothing $P(t'|X)\,P(t''|X,t')$ reduces to smoothing $P(t''|X,t')$. To obtain a correction factor $\alpha(X,t')$ that is sensitive to the history t', we can define a second back-off formula as follows

$$P_{bo}(t''|X,t') = \begin{cases} P_{\mathrm{SGT}}(t''|X,t') & \text{if } c(t',t'') = c(t) > 0, \\ \alpha(X,t')\,P(t''|Y) & \text{otherwise} \end{cases} \tag{5.2}$$

where P_{SGT} is Simple Good-Turing estimator described in Chapter 3. We want to derive $\alpha(X,t')$ such that the condition below holds:

$$\sum_{t'':c(t',t'')=0} P_{bo}(t''|X,t') \quad + \sum_{t'':c(t',t'')>0} P_{bo}(t''|X,t') = 1$$

By the definition of the back-off function, we have:

$$\sum_{t'':c(t',t'')=0} \alpha(X,t')\,P(t''|Y) \quad + \sum_{t'':c(t',t'')>0} P_{\mathrm{SGT}}(t''|X,t') = 1$$

Since $\alpha(X,t')$ does not depend on t'', we can isolate it from the summation. In addition, we can take the summation over the complementary event $t'' : c(t',t'') > 0$. Rearranging the terms, we obtain:

$$\alpha(X,t') = \frac{1 - \sum_{t'':c(t',t'')>0} P_{\mathrm{SGT}}(t''|X,t')}{1 - \sum_{t'':c(t',t'')>0} P(t''|Y)}$$

This α value can then be multiplied by $P(t'|X)$ in Equation 5.1, yielding

$$P_{bo}(t|X) = \begin{cases} P_{\mathrm{SGT}}(t|X) & \text{if } c(t',t'') = c(t) > 0, \\ P(t'|X)\,\alpha(X,t')\,P(t''|Y) & \text{otherwise} \end{cases} \tag{5.3}$$

This correction value is tighter than the one we implemented, since it is conditioned on an entire subtree. This guarantees that probability mass will flow from these subtrees strictly to their back-offs and not to all other subtrees with the same number of words. Every back-off subtree would have a different $\alpha(X,t')$ value and not only the average factor $\alpha(X,i)$. In this approach, a subtree t with i lexical items is regarded as a joint distribution of t' and all t'' that co-occurred with t' in the training set. Backing-off in this way would have the same effect as in n-gram back-off smoothing: making room for new t'' subtrees so that new joint events t', t'' could have their probabilities estimated. Because t', t'' is seen as a joint event in a joint distribution, we would have $\sum_{t''} P(t',t''|X) = P(t'|X)$. The fact that distributions conditioned on entire subtrees are so sparse, however, prevented us from following this direction.

Appendix B

Results from the OVIS treebank

All the results reported here refer to test sets with sentences having more than one word. The parameter settings follow the definitions given in Chapter 4.

Depth	1	2	3	4	5	6	7
	BO-DOP1						
Recall	95.18	95.68	95.79	95.86	96.54	96.54	96.54
Precision	91.97	93.13	93.46	93.50	94.16	94.16	94.16
Match	78.84	84.35	84.98	85.29	86.54	86.70	86.70
Av.Subst.	8.12	4.75	3.23	3.16	3.11	3.11	3.10
	DOP1						
Recall	95.18	95.58	95.72	95.86	95.93	95.93	95.94
Precision	91.97	93.06	93.39	93.50	93.57	93.57	93.57
Match	78.84	84.19	84.66	84.98	84.98	84.98	84.98
Av.Subst.	8.12	4.35	2.79	2.69	2.72	2.73	2.72
	Bon-DOP						
Recall	95.26	95.47	95.54	95.68	95.76	95.76	95.76
Precision	92.01	92.76	93.09	93.29	93.29	93.29	93.29
Match	79.00	83.10	84.19	84.51	84.35	84.35	84.35
Av.Subst.	8.12	4.31	3.19	2.97	2.94	2.94	2.94
	BO-Bon-DOP						
Recall	95.26	95.29	95.54	95.61	95.58	95.65	95.68
Precision	92.01	92.62	92.99	93.09	92.99	93.06	93.10
Match	79.00	82.32	83.41	83.72	83.41	83.57	83.57
Av.Subst.	8.12	3.93	3.06	2.94	2.93	2.92	2.92
Subtrees	2844	13088	44162	103239	149898	169404	172050

Table 5.1: Semantic, l2n4, MPD

Depth	1	2	3	4	5	6	7
	BO-DOP1						
Recall	94.47	95.44	96.34	95.83	96.27	95.98	95.98
Precision	92.20	93.17	94.27	93.78	94.14	93.86	93.86
Match	82.16	83.15	87.24	85.98	86.61	86.14	86.30
	DOP1						
Recall	94.47	95.30	96.16	95.87	96.01	95.91	95.98
Precision	92.20	93.09	94.16	93.82	93.92	93.82	93.89
Match	82.16	82.99	86.61	85.83	85.98	85.83	85.83
	Bon-DOP						
Recall	94.58	95.19	96.62	95.62	95.73	95.80	95.76
Precision	92.27	92.95	93.54	93.44	93.51	93.58	93.55
Match	82.32	82.68	84.88	84.88	84.88	85.04	85.04
	BO-Bon-DOP						
Recall	94.58	95.08	95.51	95.40	95.44	95.51	95.51
Precision	92.27	92.78	93.37	93.20	93.20	93.27	93.27
Match	82.32	81.89	84.25	83.78	83.78	83.94	83.94

Table 5.2: Semantic, l7n2L3, MPD

Depth	1	2	3	4	5	6	7
	BO-DOP1						
Recall	94.55	95.01	96.55	96.27	96.45	96.30	96.27
Precision	92.17	92.78	94.45	94.17	94.34	94.17	94.14
Match	82.32	82.68	87.56	86.77	87.09	86.61	86.46
	DOP1						
Recall	94.55	94.97	96.55	96.30	96.37	96.16	96.16
Precision	92.17	92.84	94.48	94.20	94.24	94.06	94.06
Match	82.32	82.99	87.40	86.77	86.61	86.30	86.30
	Bon-DOP						
Recall	94.58	94.72	95.80	95.69	95.91	95.94	95.94
Precision	92.21	92.59	93.71	93.54	93.72	93.79	93.75
Match	82.32	82.68	84.57	84.25	84.88	85.20	85.04
	BO-Bon-DOP						
Recall	94.58	94.72	95.52	95.69	95.87	95.91	95.91
Precision	92.21	92.40	93.47	93.57	93.68	93.72	93.72
Match	82.32	81.73	84.09	84.41	84.88	85.04	85.04
Subtrees	2571	8846	25595	60681	95111	119863	133308

Table 5.3: Semantic, l7n2L3, MPP (sample: 5000)

Split	1	2	3	4
	BO-DOP1			
Recall	96.34	96.41	96.72	95.91
Precision	94.27	93.89	94.19	93.62
Match	87.24	86.53	87.62	87.02
	$\overline{X} = 87.10$		$s = 0.4550$	
	DOP1			
Recall	96.16	96.29	96.64	95.91
Precision	94.16	93.78	94.23	93.62
Match	86.61	86.22	86.64	86.71
	$\overline{X} = 86.54$		$s = 0.2207$	
	Bon-DOP			
Recall	96.62	96.03	96.20	95.61
Precision	93.54	93.59	93.87	93.47
Match	84.88	85.14	85.83	86.40
	BO-Bon-DOP			
Recall	95.51	96.10	96.20	95.65
Precision	93.37	93.66	93.83	93.50
Match	84.25	85.14	85.34	86.40
Subtrees	25595	25841	25717	25445
Sentences	635	646	614	647
Coverage	94.21	94.03	94.02	93.36

$\overline{X}$: exact match mean; s: standard deviation

Table 5.4: Semantic, l7n2L3d3, MPD

Depth	4	5	6
	BO-DOP1		
Recall	96.15	96.22	96.08
Precision	93.74	93.81	93.67
Match	85.76	85.45	85.45
	DOP1		
Recall	95.93	96.01	96.04
Precision	93.57	93.67	93.70
Match	85.13	85.13	85.13
	Bon-DOP		
Recall	95.76	95.86	95.86
Precision	93.29	93.37	93.37
Match	84.51	84.66	84.82
	BO-Bon-DOP		
Recall	95.68	95.72	95.76
Precision	93.16	93.16	93.20
Match	84.19	84.04	84.19

Table 5.5: Semantic, l5n5, MPD

Depth	4	5	6
	BO-DOP1		
Recall	96.36	96.43	96.09
Precision	93.95	94.02	93.71
Match	86.07	85.92	85.58
	DOP1		
Recall	96.18	96.40	96.27
Precision	93.78	94.02	93.88
Match	85.60	85.92	85.58
	Bon-DOP		
Recall	95.83	95.97	95.99
Precision	93.27	93.41	93.44
Match	84.35	84.35	84.64
	BO-Bon-DOP		
Recall	95.90	96.08	96.02
Precision	93.37	93.51	93.48
Match	84.51	84.66	84.95
Subtrees	215509	560691	993483

Table 5.6: Semantic, l5n5, MPP (sample: 5000)

Depth	1	2	3	4	5	6	7
	BO-DOP1						
Recall	95.18	95.68	95.86	96.50	96.01	96.61	96.08
Precision	91.97	93.13	93.50	94.09	93.60	94.19	93.67
Match	78.84	84.35	85.29	86.07	85.45	86.38	85.60
N.Subst.	5185	3030	1837	1655	1677	1659	1657
Av.Subst.	8.12	4.74	2.87	2.58	2.62	2.59	2.59
	DOP1						
Recall	95.18	95.58	95.79	95.86	96.08	96.04	96.08
Precision	91.97	93.06	93.46	93.50	93.74	93.70	93.74
Match	78.84	84.19	84.98	84.98	85.45	85.45	85.45
N.Subst.	5185	2781	1719	1361	1357	1378	1382
Av.Subst.	8.12	4.35	2.69	2.12	2.12	2.15	2.16
	Bon-DOP						
Recall	95.26	95.40	95.58	95.76	95.90	95.90	95.90
Precision	92.01	92.69	93.12	93.29	93.40	93.40	93.40
Match	79.00	82.79	84.19	84.51	84.66	84.82	84.82
N.Subst.	5186	2750	1956	1606	1563	1563	1565
Av.Subst.	8.12	4.30	3.06	2.51	2.44	2.44	2.44
	BO-Bon-DOP						
Recall	95.26	95.36	95.58	95.68	95.76	95.83	95.79
Precision	92.01	92.69	93.09	93.13	93.16	93.23	93.20
Match	79.00	82.63	83.72	84.04	84.04	84.19	84.19
N.Subst.	5186	2491	1865	1629	1598	1586	1587
Av.Subst.	8.12	3.89	2.91	2.54	2.50	2.48	2.48
Subtrees	2844	13329	48289	143229	254018	330248	360399
Sentences	638	639	639	639	639	639	639
Coverage	94.65	94.80	94.80	94.80	94.80	94.80	94.80

Table 5.7: Semantic, l3n4, MPD

Depth	1	2	3	4	5	6	7
	BO-DOP1						
Recall	90.34	93.56	95.25	95.62	95.78	95.65	95.76
Precision	85.67	90.79	92.73	93.17	93.31	93.21	93.39
Match	67.96	77.99	81.89	83.38	83.98	83.38	83.81
N.Subst.	5646	2759	1831	1531	1569	1587	1576
Av.Subst.	8.45	4.13	2.74	2.29	2.34	2.37	2.36
	DOP1						
Recall	90.34	93.76	95.52	95.85	95.95	95.78	95.96
Precision	85.67	90.98	93.05	93.52	93.71	93.49	93.74
Match	67.96	78.74	82.49	83.68	83.98	83.53	83.81
N.Subst.	5646	2750	1705	1263	1305	1341	1346
Av.Subst.	8.45	4.11	2.55	1.89	1.95	2.00	2.01
	Bon-DOP						
Recall	90.11	93.89	95.45	95.52	95.85	95.78	95.73
Precision	85.19	91.05	92.86	92.84	93.10	93.06	93.12
Match	67.66	77.69	82.34	81.89	82.49	82.49	82.76
N.Subst.	5659	2781	1719	1350	1356	1355	1353
Av.Subst.	8.47	4.16	2.57	2.02	2.02	2.02	2.02
	BO-Bon-DOP						
Recall	90.11	93.66	95.32	95.32	95.75	95.72	95.63
Precision	85.19	90.71	92.61	92.61	92.91	92.94	92.99
Match	67.66	77.10	81.59	81.14	81.74	81.89	82.01
N.Subst.	5659	2769	1704	1452	1452	1456	1455
Av.Subst.	8.47	4.14	2.55	2.17	2.17	2.17	2.17
Subtrees	1626	8756	37255	121031	221543	291265	319526
Sentences	668	668	668	668	668	668	668
Coverage	99.10	99.10	99.10	99.10	99.10	99.10	99.10

Table 5.8: Syntactic, l3n4, MPD

Bibliography

Abney, Steven (1987). The English Noun-Phrase and its sentential aspect. Ph.D. diss., MIT, Cambridge, Massachusetts.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities* (3): 1–8.

Black, E., S. Abney, D. Flickinger, C. Gdaniec, 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 grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop.*

Bod, Rens (1992). Data oriented parsing (DOP). In *Proceedings COLING 1992*, Nantes, France.

Bod, Rens (1993). Using an annotated corpus as a stochastic grammar. In *Proceedings EACL 1993*, Utrecht, The Netherlands.

Bod, Rens (1996). Two questions about data-oriented parsing. In *Proceedings COLING 1996*, Copenhagen, Denmark.

Bod, Rens (2000). Parsing with the shortest derivation. In *Proceedings COLING 2000*, Saarbrücken, Germany.

Bod, Rens (2001). What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings ACL 2001*, Toulouse, France.

Bod, Rens and Ronald Kaplan (1998). A probabilistic corpus-driven model for lexical functional analysis. In *Proceedings COLING-ACL 1998*, Montreal, Canada.

Bonnema, Remko, Rens Bod, and Remko Scha (1997). A DOP model for semantic interpretation. In *Proceedings COLING-EACL 1997*, pp. 159–167, Madrid, Spain.

Bonnema, Remko, Paul Buying, and Remko Scha (1999). A new probability model for data-oriented parsing. In Dekker, P., Kerdile G., editor, *Proceedings of the 12th Amsterdam Colloquium*, Amsterdam, The Netherlands.

Bonnema, Remko and Remko Scha (2002). Reconsidering the probability model for data-oriented parsing. In Rens Bod, Remko Scha and Khalil Sima'an, editors, *Data-Oriented Parsing*, pp. 23–40. CSLI Publications.

Brants, Thorsten and Matthew Crocker (2000). Probabilistic parsing and psychological plausibility. In *Proceedings COLING 2000*, Saarbrücken / Luxembourg / Nancy.

Carroll, John and David Weir (2000). Encoding frequency information in lexicalized grammars. In Bunt, H. and A. Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*. Kluwer Academic Publishers.

Charniak, Eugene (1999). A maximum-entropy-inspired parser. Technical report 12, Brown University, Providence, Rhode Island.

Chater, Nicholas (1996). Reconciling simplicity and likelihood principles in perceptual organization. *Psychological Review* (103): 566–581.

Chater, Nicholas, Matthew Crocker, and Martin Pickering (1998). The rational analysis of inquiry: The case for parsing. In Chater and Oaksford, editors, *Rational Models of Cognition*. Oxford University Press.

Chen, John and K. Vijay-Shanker (2000). Automated extraction of tags from the penn treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*, Trento, Italy.

Chen, Stanley F. and Joshua Goodman (1998). An empirical study of smoothing techniques for language modeling. Technical report 10, Harvard University, Cambridge, Massachusetts.

Chomsky, Noam (1957). *Syntactic structures*. Mouton, The Hague.

Church, Kenneth and Ramesh Patil (1982). Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics* (8): 139–149.

Collins, Michael (1997). Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the EACL*, pp. 16–23, Madrid, Spain.

Crocker, Matthew (1996). Mechanisms for sentence processing. Technical report 70, University of Edinburgh, Edinburgh, UK.

Gale, William and Kenneth Church (1994). What is wrong with adding one? In Oostdijk, Nelleke and Pieter Haan, editors, *Corpus-based research into language*, Amsterdam, The Netherlands. Rodopi.

Gale, William and Geoffrey Sampson (1995). Good-turing frequency estimation without tears. *Journal of Quantitative Linguistics* (2): 217–237.

Goodman, Joshua (1998). Parsing Inside-Out. Ph.D. diss., Harvard University, Cambridge, Massachusetts.

Hoogweg, Lars (2000). Extending dop1 with the insertion operation. Master's thesis, University of Amsterdam, Amsterdam.

Jelinek, Frederick (1999). *Statistical Methods for Speech Recognition.* MIT Press.

Jelinek, Frederick and Robert L. Mercer (1980). Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands.

Johnson, Mark (2002). The DOP estimation method is biased and inconsistent. *Computational Linguistics* 28(1): 71–76.

Katz, Slava M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, And Signal Processing* 35(3): 400–401.

Lombardo, Vincenzo and Patrick Sturt (2002). Incrementality and lexicalism: a treebank study. In Stevenson, S. and P. Merlo, editors, *Lexical Representations in Sentence Processing*, Computational Psycholinguistics Series, pp. 137–155. John Benjamins.

Magerman, David (1995). Statistical decision-tree models for parsing. In *Proceedings of ACL 1995*, Cambridge, Massachusetts.

Manning, Christopher D. and Hinrich Schütze (2001). *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts.

Marslen-Wilson, William (1973). Linguistic structure and speech shadowing at very short latencies. *Nature* (244): 522–533.

Sampson, Geoffrey (2001). *Empirical Linguistics.* Continuum International, London.

Scha, Remko (1990). Language theory and language technology; competence and performance (in dutch). In de Kort, Q. and G. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, Almere. Landelijke Vereniging van Neerlandici (LVVN-jaarboek). For an English version, see http://www.iaaa.nl/rs/LeerdamE.html.

Sima'an, Khalil (1999). *Learning Efficient Disambiguation.* ILLC Dissertation Series, 1999-02. University of Amsterdam/Utrecht University, The Netherlands.

Sima'an, Khalil (2001a). Enhancing the robustness of data-oriented parsing for speech-understanding. In *Proceedings of the International Workshop on Parsing Technologies*, Beijing, China.

Sima'an, Khalil (2001b). Tree-gram parsing: Lexical dependencies and structural relations. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS'01)*, Tokyo, Japan.

Srinivas, Bangalore and Aravind Joshi (1999). Supertagging: an approach to almost parsing. *Computational Linguistics* 25(2): 237–266.

van Zanten, Gert Veldhuizen (1996). Semantics of update expressions. Technical report 24, NWO Priority Programme Language and Speech Technology, The Hague, The Netherlands.

Weischedel, R., M. Meteer, R. Schwarz, L. Ramshaw, and J. Palmucci (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics* (2).

Wonnacott, Thomas and Ronald Wonnacott (1990). *Introductory Statistics.* John Wiley and Sons, USA.