Chapter 2

The task of parse selection

2.1 Syntax and structure in language

In modern linguistic theory and practical natural language processing (NLP) the assumption is made that human language is structured and that understanding an utterance requires that aspects of its underlying structure be determined. The specification of this structure often takes the form of some kind of generative grammar, by means of which words of the language may be combined using syntactic rules. The notion of such a grammar in contemporary linguistics dates back to Chomsky (1957), although startlingly similar techniques were used by the Hindu scholar Pāṇini in his grammar of Sanskrit in approximately the 5th century BC (Ingerman, 1967). In contemporary notation, a grammar rule, or a rewrite rule states that what is on its left side may be composed of (or rewritten as) the concatenation of symbols on its right side. A symbol in a grammar which may be rewritten or expanded in this way is called a non-terminal. Successive applications of rules to derive a string may be visualized as parse trees. In order for the meaning of a sentence to be retrieved once the syntactic structure is known, relationships between syntactic and semantic structures must be defined. In some conceptions of grammar, such as Montague grammar (Montague, 1973), syntax and semantics are considered to be encoded together, whereas in others, some mapping between syntactic structures and semantic relations is defined. The syntactic structure of a sentence then yields information about the relationships between its semantic components.

A simple example of how a difference in syntactic structure will yield a different semantic reading may be seen in example 2.1. In this example, the two different syntactic structures differ in prepositional phrase attachment. That is to say, in the first example, the prepositional phrase “with a telescope” is “attached” to the noun phrase (NP) node of the tree, whereas in the second it is attached to the verb phrase (VP) node of the tree. The semantic interpretation of these various prepositional phrase attachments is intuitive. The first indicates
Figure 2.1: Two parse trees with different PP attachments for the sentence *The peeping-tom saw the nudist with a telescope*.

that the full NP is “the nudist with a telescope”, whereas the second has the VP “saw ... with a telescope”.

The two structures shown in 2.1 are parse trees. This example also serves to illustrate the fact that a single sentence may have more than one underlying syntactic structure. This is what is meant by the term *syntactic ambiguity* in language. In later sections it will become clear that ambiguity of much greater proportions than this is the norm.

As discussed before, it is a common assumption that in order for a sentence to be completely understood, its correct syntactic structure must be retrievable. The process of retrieving this structure, given a sentence and a grammar, is *parsing*.

### 2.1.1 Dependency structures

Another approach to grammar analysis focuses more on the relations between words in a sentence than on the construction of trees of syntactic constituents. In *dependency grammars*, each word is considered to be dependent upon another word. The individual dependencies are labeled with the relation between the two words. Part of the idea of such a grammar is to describe relations between constituents which do not depend upon the constituents’ positions in the sentence. For this reason, languages such as Czech, where constituent order is relatively
free, are often better analyzed using some variety of dependency grammar, such as is used in the 500,000 word Prague Dependency Treebank (Hajic, 1998; Collins et al., 1999). In chapter 7, dependency structures will contribute information about the parse structure sought.

### 2.2 The basics of parsing

The task of parsing is essentially one of searching through a space of possible trees to find those which, beginning from a starting non-terminal S, will yield the sentence being parsed through repeated expansions of rules in the grammar. This search can be conducted in a top down fashion, or bottom up, or, by use of a technique called *filtering*, a combination of the two.

#### 2.2.1 Earley parsing

One widely used algorithm for parsing is the Earley algorithm (Earley, 1970), a top-down, left to right parsing algorithm which uses a chart to record rules which have been applied at each stage of the derivation.

Variations of this algorithm have been used in statistical parsers such as the *Pearl* parser of Magerman and Marcus (1991). An informal description of the algorithm here will serve as an illustrative example of the procedure of parsing. A more in-depth look at this and other parsing methods (as well as an admirable introduction to the field of NLP in general) can be found in Jurafsky and Martin (2000).

The Earley algorithm is a method of parsing context-free grammars (CFGs). Many of the grammars used in NLP are more expressive than CFGs, as is the case of so-called attribute-value or unification-based formalisms such as Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) or Lexical
CHAPTER 2. THE TASK OF PARSE SELECTION

\[
\begin{align*}
\text{<ROOT>} & \rightarrow \text{main:} \\
\text{main:} & \rightarrow \text{subj:} \text{ saw } \text{obj:} \\
\text{subj:} & \rightarrow \text{John} \text{ Sara} \\
\text{obj:} & \rightarrow \text{mod:} \\
\text{mod:} & \rightarrow \text{with} \\
\text{attr:} & \rightarrow \text{friend} \\
\text{punct:} & \rightarrow \text{best} \\
\end{align*}
\]

Figure 2.3: An example of a dependency relation structure, drawn in the form of a tree. The labels on edges refer to the relations of subject, object, modifier, prepositional complement, and premodifying (attributive) nominals.

\[
\begin{align*}
S & \rightarrow \text{NP VP} & \text{Det} & \rightarrow \text{a|the} \\
\text{NP} & \rightarrow \text{Verb NP} & \text{Verb} & \rightarrow \text{reads} \\
\text{VP} & \rightarrow \text{Verb} & \text{Noun} & \rightarrow \text{man|book} \\
\text{NP} & \rightarrow \text{Det Noun} & \\
\end{align*}
\]

Figure 2.4: A miniature grammar and lexicon

Functional Grammar (LFG) (Kaplan and Bresnan, 1982). Still others lie somewhere between, such as Definite Clause Grammars (DCGs), such as the Tag Sequence Grammar (Briscoe and Carroll, 1993) used in the experiments described in chapter 6. In some cases, when parsing certain non-context free grammars such as LFG, it is a useful tactic to extract a context-free backbone, which can be embellished with further information in order to facilitate parsing, as discussed in practical terms recently in Dowding et al. (2001). Thus context-free parsing algorithms such as Earley can be useful even when dealing with more powerful descriptive formalisms.

The Earley algorithm involves a left to right pass over the sentence to be parsed which is \( N \) tokens long. For each word in the sentence (plus an initial entry at position zero), an entry is made into the chart which consists of a list of states representing possible partial parse structures which can be constructed with the words seen up until that point. A state consists of a single grammar rule which is notated in such a way (usually by use of a dot) to describe how much of the right hand side of that rule has been instantiated thus far, as well as an
2.2. **The Basics of Parsing**

index of the span of the rule over word positions in the sentence so far. At the beginning of the parsing process, for example, a state in the initial chart entry for a grammar like the one in table 2.4 would look like this:

\[(2.1) \quad S \rightarrow \bullet NP \ VP, [0, 0] \]

The rule used is \( S \rightarrow NP \ VP \). The dot to the left of the \( NP \) indicates that none of the rule has yet been expanded, and the index \([0, 0]\) indicates that none of the words have yet been seen.

At the end of the pass, the states in the chart constitute a record of all the possible parses of the sentence. The process of adding states and updating entries is the main task of the algorithm. This is done by the use of three operators, called the **Predictor**, the **Scanner**, and the **Completer**. The **Predictor**’s function is to create new states in the current chart entry. It does this by considering any non-terminals (which are not part-of-speech categories) to the right of a dot in all states in the current entry, and looking up grammar rules which have such a non-terminal as their left hand side. These rules are then used as new states in the same chart entry. After this step in the example with the grammar in table 2.4 the following state is added to the initial entry:

\[(2.2) \quad NP \rightarrow \bullet Det \ Noun, [0, 0] \]

The next step involves the use of the **Scanner**, which reads the next word of input and searches for corresponding part-of-speech predictions in the current entry’s states. If we imagine an example sentence **the man reads**, then this operation will read the word **the** and add a new state to the **following** chart entry corresponding to the state in 2.2 with its dot and index updated accordingly:

\[(2.3) \quad NP \rightarrow Det \bullet Noun, [0, 1] \]

When a state is reached where the dot has moved to the end of a rule, indicating that a full constituent has been seen, the **Completer** is employed to update previous states which had been waiting for the category which has been filled. Thus, in our example, when the next word, **man** is read,

\[(2.4) \quad NP \rightarrow Det \ Noun\bullet, [0, 2] \]

is processed, the completer looks for states whose index ends in 0 (indicating that this is the position in the sentence where they were left off) which need an \( NP \) to advance. In this case, the state in 2.1 is what it finds, and this state is updated to

\[(2.5) \quad S \rightarrow NP \bullet VP, [0, 2] \]

The use of these three operators continues until all of the words of the sentence have been seen and all the possible states which can be made have been entered
into the chart. By including in each state created by the Completer information pointing back to a list of its predecessor states, all the possible parse structures of the sentence can be retrieved from the chart.

2.3 Ambiguity and large grammars

Techniques such as Earley parsing are fine for simple recognition tasks, where it is only important to know that there is some grammatical parse. If there is some state \( S \rightarrow \alpha \cdot, [0, N] \) in the last entry of the chart, then the sentence is a grammatical string. Since all possible parses can be extracted from the finished chart, the question of whether there is any parse is easily answered. The problem arises, however, when there are many parses, and it is necessary to retrieve the correct one. The presence of multiple parses given a sentence and a grammar is what has been introduced already as ambiguity, and to resolve it requires a more sophisticated approach than simple chart parsing. Chart parsing alone will tell us what structures are licensed, but it will not tell us which of the structures is preferable.

As discussed in section 1.2, the problem of ambiguity in large grammars is a well-known stumbling block in practical NLP (Church and Patil, 1982). Hindle & Rooth (1993) give the following example:

\[
[\text{np The space}] [\text{vpres probes}] [\text{np detected lightning}] [\text{pp in Jupiter’s upper atmosphere}] \text{ and observed auroral emissions like Earth’s northern lights in the Jovian polar regions.}
\]

As may be seen by the partial parse assigned to the bracketed portion of the sentence, word and part of speech ambiguities lead to structural assignments which are clearly not acceptable to human parses. As figure 1.2 in chapter 1 showed, the number of possible structures grows exponentially in relation to the length of the sentence.

The problem is that the resources used by humans to assess the correctness of a parse are often of an “extra-grammatical” nature. Semantics play a crucial role, as does real world knowledge. For example, the earlier sentence example the peeping tom saw the nudist with a telescope exhibits ambiguity in the form of the two parses illustrated in figure 2.1. In this case, both possible parses are reasonable, since we know that a telescope can be a tool to aid seeing. An analogous sentence such as the peeping tom saw the nudist with a sunburn, seen in figure 2.2 does not appear to a human to have the same ambiguity, since to attach the PP with a sunburn to the verb would make no sense. Our real-world knowledge of the nature of sunburns and telescopes resolves this potential ambiguity. However, this difference is not reflected in the grammar. A computational grammar generally is very limited in the amount of such knowledge it has at its disposal, and it is clear that to attempt to incorporate extensive amounts of such information
into a grammar would lead to considerable difficulties, as the line separating "linguistic" knowledge from all other forms of cognition becomes increasingly blurred. Whether, given this lack of distinction, the notion of a "grammar" would continue to have any real meaning whatsoever is a separate issue. It is certainly a moot point for current NLP; it is simply not feasible to incorporate into computational grammars the kind of massive quantities of real world knowledge which would be necessary for purely symbolic disambiguation, even if it were certain that such real world knowledge would even be sufficient.

The ambiguity increases with the coverage of the grammar. The more varied the structures are that the grammar can admit, the more these structures will be likely to appear in inappropriate places. The task of producing all parses of a string using a large attribute-value grammar has a worst-case complexity which is exponential on the length of the string. Even though, as Carroll (1994) points out, the worst-case complexities are rarely reached in practice, the proliferation of bad parses for any sentence of significant length necessitates the use of disambiguation techniques. For sentences of length 28 to 30 words, Carroll (1994) reports a mean number of parses produced by the Alvey NL Tools Grammar of 343.5, with a standard deviation of 693.7. The maximum number of parses he reports was 2736 for one 29-word sentence, while many sentences of the same length yielded fewer than ten parses. In spite of the fact that there are many sentences with a relatively low level of ambiguity, the mean number of parses increases quickly with respect to the length of the sentence. The same paper reports that for 25-27 word sentences, the mean number of parses produced by the same grammar was 168.8, with a standard deviation of 303.1.

These numbers are largely dependent on the coverage of the specific grammar used and the domain of the sentences being parsed. As mentioned in chapter 1, the Alpino grammar of Dutch produced in an extreme case over 500,000 unique parses for a sentence of only 16 words.

As mentioned above, the resources available to a human for disambiguation are varied, and often of a non-linguistic nature. To imitate the depth of semantic comprehension and real-world knowledge employed by a human is, at present, out of the question for computational NL applications. Instead, contemporary NL disambiguation has moved in the direction of statistical methods, borrowing many strategies from the related discipline of speech recognition.¹ The remainder of this work will focus on the task of finding the correct parses of sentences by means of statistical methods.

¹For a solid introduction to the statistical methodologies employed in contemporary NLP, both Manning and Schütze (1999) and Jelinek (1998) are valuable foundational texts.
2.3.1 Statistical parsing and “parse selection”

In this work the terms “statistical parsing” and “parse selection” are both used, sometimes interchangeably. It is worthwhile to clarify the distinction for cases in which it makes a difference. Generally speaking, I will use the former term to refer to parsing which has statistical elements integrated in such a way that its output is a parse or parses which are ideally the best possible. A less integrated way to go about the task is to have a non-statistical parser whose job it is simply to create all possible parses in advance (perhaps in the form of a parse forest, a compact representation of multiple parse trees), at which point a separate statistical model is employed to select the best of those possibilities. This approach I refer to as “parse selection”. For reasons I will elaborate on later in this work, the specific experiments of which I am presenting results in chapter 6 were done in this way, whereas in chapter 7, an integrated approach is employed in the form of a beam search of a parse forest, in which the model is used to rank potential subtree expansions. It is worth noting that in such an approach, the most probable subtree at any given point within the search is not necessarily the correct one, from the perspective of the whole sentence. The width of the beam determines the degree of locality of this, as will be discussed further in chapter 7.

2.4 Performance evaluation

The goal of statistical parsing as suggested so far is not well defined. As is clear from the example in figure 2.1, there are cases in which neither parse is clearly bad. By definition, all parses are admissible grammatical interpretations of the sentence, but there is a broad range possible of degrees of preference for or against a parse. For this reason, it is necessary to arrive at some meaningful scheme of evaluating parses.

The standard approach to this is to use a corpus of hand-parsed (or hand-selected) data. The handmade parse is then treated as a gold standard against which automatic parses of the same sentence are compared. This simplifies the problem. It makes the assumption that for every sentence there is a single best parse, and this is the parse created by a human parser. Usually, this is the case. It is possible, however, that a sentence might have two or more parses which are equally acceptable to a human parser, as in the case of the sentence in example 2.1. It is also possible that in some cases individuals will make different judgements on plausibility. Cases such as these are treated as noise.

The next, and more serious, compromise to be made in order to define the problem is in determining a distance metric to evaluate the closeness of the automatic parses to the handmade gold standard. The current standard approach to this incorporates several measurements (called the PARSEVAL measures) proposed by Black et al. (1991), most significantly those of labeled precision, labeled
2.5. **Approaches to Statistical Grammars**

Recall, and crossing brackets. These measurements reflect different ways in which an evaluated parse might diverge from a gold standard parse. Labeled precision is the number of correct constituents in the evaluated parse of \( s \) divided by the total number of constituents in the evaluated parse of \( s \). Labeled recall is the number of correct constituents in the evaluated parse of \( s \) divided by the number of correct constituents in the gold standard parse of \( s \). The crossing brackets measure is the number of constituents in which the brackets of the evaluated parse and the gold standard parse overlap, such as in the case of \((AB)C\) and \((A(BC))\), where the bracketings cross over the B. A freely distributed implementation of these measurements is the evalb software of Sekine and Collins (1997). Precision and recall scores are often combined to yield a single score, called an f-score, which is calculated as follows:

\[
    f = \frac{1}{\alpha \frac{p}{r} + (1 - \alpha) \frac{r}{p}}
\]

with \( \alpha \) often set to equal 0.5, representing an equal weighting of precision and recall, yielding the equation

\[
    f = \frac{2pr}{p + r}
\]

where \( f \) is the f-score and \( p \) and \( r \) are precision and recall, respectively.

The advantages of using clear metrics are obvious in terms of defining the task. These advantages are not without a cost, however. The PARSEVAL measures, while very useful, do not always correspond to human intuitions about preference of parses. They are only an approximation of the true measurement which is sought, that of "closeness" of parses, as judged by human native speakers, but they are commonly used measurements. Other metrics might also be used, such as comparing dependency relations, as will be discussed further in chapter 7. In chapter 6, a simpler measurement of performance is employed, for reasons discussed there, of simply evaluating how many of the sentences the model was able to choose the single best parse for. This approach tends to lead to apparently worse results, since "near misses" are counted as unsuccessful, but it proves to be sufficient for the purposes of chapter 6. Chapter 7 will also introduce the metric of concept accuracy, and a general purpose score, referred to as phi which allows results to be compared even when the exact task is slightly different, as is often the case in a rapidly developing grammar environment such as the Alpino environment discussed in that chapter.

2.5 Approaches to statistical grammars

This section describes various approaches to statistical grammars and indicates the current state of the art with regard to performance.
2.5.1 PCFGs

The simplest way to incorporate a statistical element in the parsing of context-free languages is to build the probabilities directly into the grammars themselves. Such grammars are called probabilistic (or stochastic) context-free grammars (PCFGs) (Booth and Thompson, 1973). In a PCFG, each rule is associated with some probability. The sum of the probabilities associated with all rules which share a left side is equal to one. This means that there is a proper probabilistic distribution over all possible expansions of any non-terminal. In this case, the probability of a given parse is equal to the product of the probabilities of each expansion used in the parse.

PCFGs are simple, and the probability of a parse is easy to determine. They assume that the most probable parse is that which is made up of the most probable expansions at each non-terminal point in the derivation. An Earley-style parser can be easily modified to incorporate the probabilistic information. However, PCFGs fail in several important respects.

The PCFG independence assumption

The first problem which PCFGs have is their inability to represent non-local structural dependencies within parses. Recalling the examples in fig 2.1 and 2.2, it is clear that the likelihood of the $VP \rightarrow NP \ PP$ rule application is statistically dependent on which rule application is used in connection with the $PP$, e.g., whether the $PP$ can describe a manner of seeing. In PCFGs, no such dependencies can be modeled. Since the probabilities of all expansion rules used in the parse are simply multiplied together to yield the probability of the full parse, even more explicitly structural statistical dependencies, if they are not sufficiently local, are not reflected in the final probability. It is of course imaginable (and in fact generally considered to be the case) that a language might have structures within it which are more or less likely to occur in the presence of other structures. Such tendencies cannot be modeled using an ordinary PCFG. This independence assumption has other repercussions as well, since contemporary generative linguistics and NLP often make use of more expressive grammatical formalisms than CFGs to describe language which rely upon statistically dependent structures and elements for their expressive power. Attribute-value (AV) grammars, for example, cannot be accurately modeled using PCFG-style techniques, since they contain statistical dependencies in the form of values shared by multiple features (Abney, 1997). Such shared values arise in many instances in AV grammars, for example in representing agreement between subjects and verbs. Counts of CF rule expansions in which this agreement information is written into the rules, will not reflect the fact that one rule’s expansion is statistically dependent upon how another rule was expanded, and probabilities derived from such counts will be inaccurate.
2.5. APPROACHES TO STATISTICAL GRAMMARS

Lexical dependencies

PCFGs attach probabilities to each expansion in a parse. Usually, the process of assigning POS tags to lexical items in the sentence is done in advance, and the PCFG considers as its terminals the POS tags themselves. Several early parsers treated POS tags as terminals in this way, including the aforementioned Perl parser of Magerman and Marcus (1991). However, more recent work has recognized the importance of words in parse selection. It has been demonstrated that lexical information is crucial in deciding between certain structural alternatives, such as in the case of PP attachment (Hindle and Rooth, 1993). In the sentence from figure 2.1, for example, a statistical correlation between the occurrence of the words telescope and saw would help lead to the preferable parse. It would of course be possible to represent the tagging process in the form of unary branching probabilistic rules from POS tags to words. This does not solve the problem, however, of allowing word information to be accessible at important points in the parse tree.

2.5.2 Lexicalized parsing

It is widely accepted that lexical information in a sentence is important in disambiguation (Hindle and Rooth, 1993; Charniak, 1997; Magerman, 1993; Collins, 1999). The following parsers illustrate the advancing state of the art in lexicalized parsing throughout the last five to ten years.

Magerman

An early instance of a contemporary lexicalized parsing system is Magerman’s decision-tree based SPATTER parser (Magerman, 1993; Magerman, 1994; Magerman, 1995). SPATTER parses in a bottom up, left to right fashion by considering words as terminal nodes in a tree and adding edge extensions upward from nodes based on a decision tree mechanism which considers various contextual factors of the node in question. The edge extensions assigned may be any one of five possible types:

<table>
<thead>
<tr>
<th>term</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>right</td>
<td>the current node is the first child of the constituent to be built</td>
</tr>
<tr>
<td>left</td>
<td>the current node is the last child of the constituent</td>
</tr>
<tr>
<td>up</td>
<td>the current node is neither first nor last child of the constituent</td>
</tr>
<tr>
<td>unary</td>
<td>the current node is only child of the constituent</td>
</tr>
<tr>
<td>root</td>
<td>the current node is the root of the tree</td>
</tr>
</tbody>
</table>

A constituent is built when a right extending node matches a left extending node, with any number of intervening up nodes. Note that the terms “right” and “left” here refer to the direction of the extension viewed from the lower node. The first child of a constituent is viewed as having an extension upwards and to the right which connects to the mother node. When a left extension
Figure 2.5: An example of a constituent in SPATTER, with points of information labeled.

has been assigned, the constituent is complete. This constituent is then labeled and its parent node is extended in the same way as the lexical nodes had been extended.

The decisions of which POS tag to assign to the words, which type of edge extension to attach to a node, and which label to give a newly completed constituent are conditioned on the following factors, formulated as questions where X may mean word, tag, label, or extension and Y indicates either left or right:

- What is the X at the current node?
- What is the X at the node to the Y?
- What is the X at the node two nodes to the Y?
- What is the X at the current node's first child from the Y?
- What is the X at the current node's second child from the Y?

Figure 2.5 shows the kind of structure being built, with lexical nodes identified by their word and POS tag and higher level nodes identified by label. Note once again that “right” and “left” refer to the direction of the extension from the lower node to the upper node. The extension from the first constituent extends in the direction upwards and to the right, and upwards to the left from the last constituent.

The reports on the SPATTER system emphasize the importance of lexical and contextual information to condition decisions made in the parsing process.

Results are given for parsing of sentences in the Wall Street Journal Penn Treebank corpus (Marcus, Santorini, and Marcinkiewicz, 1993). In sentences of under 40 words in length, the SPATTER system attains a labeled recall result of 84.6%, a labeled precision result of 84.9% and an average number of crossing brackets per sentence of 1.26 (Collins, 1997).

Subsequent statistical parsers have placed similar emphasis on the usefulness of lexical statistics and contextual information. In general, the task is concerned with assigning probabilities to steps which in some way correspond
2.5. **APPROACHES TO STATISTICAL GRAMMARS**

to grammar productions. One approach to doing this is to create a *treebank grammar* (Charniak, 1996), a grammar built up of expansions collected from the treebank itself, with their probabilities determined from their empirical frequency distribution in the treebank. Such a grammar produces reliable probabilities for expansions which occur often in the data, but cannot give satisfactory probabilities for possible expansions not found in the data, or which occur too scarcely to provide sound frequency estimates. An approach to assigning probabilities to unseen rules employed by Magerman (1995) and subsequent parsers such as Collins (1997) and Charniak (1999) is that of *Markov grammars*, where an expansion is derived incrementally using probabilities conditioned on things already known about the mother node or the sister nodes. The number of previous steps in the expansion which condition the next step is referred to as the *order* of the Markov grammar.

**Collins**

In Collins (1997), a tree is derived top-down using statistics derived from a *lexicized PCFG*. This is a PCFG where rules are collected from a treebank in which each node in a tree is annotated with its own head lexical item. The resulting grammar contains a large number of rules, since each non-terminal is associated with a specific lexical entry. In order to assign a probability to a structure, a Markov grammar is used which first generates $H$, the head daughter of the constituent, conditioned on $P$ and $h$, the label and head word of the parent, respectively. Once the head daughter is determined (the lexical head $h$ is inherited from $P$), modifiers to the right of the head are generated with probability

$$
\prod_{i=1}^{m+1} \mathcal{P}_R(R_i(r_i)|P, h, H)
$$

where $R_1...R_m$ are the right modifiers of $H$ and $r_1...r_m$ are their head words. $R_{m+1}(r_{m+1})$ is defined as STOP, which prevents the model from generating any further to the right. After this, the modifiers to the left of the head are generated likewise, again with the STOP symbol demarcating the edge of the expansion, using the appropriate equation:

$$
\prod_{i=1}^{n+1} \mathcal{P}_L(L_i(l_i)|P, h, H)
$$

As an example, the values associated with the constituent tree

$S(bought, VBD) \rightarrow NP(week, NN) \ NP(IBM, NNP) \ VP(bought, VBD)$

are:

$n = 2$ \hspace{1cm} $m = 0$ \hspace{1cm} P = S

$H = VP$ \hspace{1cm} $L_1 = NP$ \hspace{1cm} $L_2 = NP$

$L_3 = \text{STOP}$ \hspace{1cm} $R_1 = \text{STOP}$ \hspace{1cm} $h = \langle \text{bought, VBD} \rangle$

$l_1 = \langle \text{IBM, NNP} \rangle$ \hspace{1cm} $l_2 = \langle \text{week, NN} \rangle$
Here, for example, \( n = 2 \) because there are two daughters to the left of the head, which is marked \( \text{VP} \). After generating the appropriate number of nodes in either direction, the parser reaches the \( \text{STOP} \) tag.

Plugging the values into the product above yields the probability equation for the left expansion:

\[
\mathcal{P}_L(\text{NP}((\text{IBM}, \text{NNP}))|S, \langle \text{bought, VBB, VP} \rangle) \\
\times \mathcal{P}_L(\text{NP}((\text{weak, NN}))|S, \langle \text{bought, VBB, VP} \rangle) \\
\times \mathcal{P}_L(\text{STOP}|S, \langle \text{bought, VBB, VP} \rangle)
\]

The probabilities for both right and left modifier generations are conditioned on only one previously generated label, that of the head daughter, thus this grammar is a 1\(^{st}\) order Markov grammar. This need not be the case. The generation of each modifier may be conditioned on those any previous generative steps. Furthermore, if the derivation is depth-first, the generation of each modifier may be conditioned on contexts which actually occur below it in the parse tree. Collins approximates these higher order Markov grammars by using a distance function:

\[
\begin{align*}
\mathcal{P}_r(L_i(l_i)|P, h, H, L_1(l_1) \ldots L_{i-1}(l_{i-1})) = \mathcal{P}_r(L_i(l_i)|P, h, H, \text{distance}_r(i - 1)) \\
\mathcal{P}_l(L_i(l_i)|P, h, H, \text{distance}_l(i - 1))
\end{align*}
\]

A similar equation is used to approximate the probabilities of right side modifiers. The distance measure is the same as in Collins (1996), a vector of three pieces of information: whether the string is of zero length, whether the string contains a verb, and whether the string contains 0, 1, 2 or > 2 commas.

A modified version of the parser (Model 2) adds probabilities over subcategorization frames. These are generated immediately after the head node is chosen and before the right and left modifiers are generated. The modifiers are then conditioned as in equation 2.8, but with the added conditioning factors \( RC \) and \( LC \), the right and left subcategorization frames, respectively:

\[
\begin{align*}
\mathcal{P}_r(R_i(r_i)|P, h, H, \text{distance}_r(i - 1), RC) \\
\mathcal{P}_l(L_i(l_i)|P, h, H, \text{distance}_l(i - 1), LC)
\end{align*}
\]

The subcategorization frames used here are simply multisets of complements required by the head. When a complement is generated by this process, the subcategorization frame is updated and the complement which was generated is removed from the multiset.

Collins’ Model 3 incorporates a final added parameter to model the percolation of gaps up a parse tree. In this model, the left side of rules for gapped
Table 2.1: Performance of Magerman, Collins, and Charniak’s parsers on sentences of \( \leq 100 \) words. The parsers are evaluated in terms of labeled recall, labeled precision, and (incorrectly) crossing brackets.

<table>
<thead>
<tr>
<th>Model</th>
<th>LR</th>
<th>LP</th>
<th>CBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.0%</td>
<td>84.3%</td>
<td>1.46</td>
</tr>
<tr>
<td>Collins 97 Model 1</td>
<td>86.8%</td>
<td>85.7%</td>
<td>1.11</td>
</tr>
<tr>
<td>Collins 97 Model 2</td>
<td>87.5%</td>
<td>88.1%</td>
<td>1.32</td>
</tr>
<tr>
<td>Collins 97 Model 3</td>
<td>87.5%</td>
<td>88.1%</td>
<td>1.07</td>
</tr>
<tr>
<td>Charniak 99</td>
<td>89.6%</td>
<td>89.5%</td>
<td>0.88</td>
</tr>
</tbody>
</table>

(2.11) \( \mathcal{P}_G(G|P, h, H) \)

where \( G \) can one of the three values \textbf{Head}, \textbf{Left}, or \textbf{Right}. In the event that the resulting value is \textbf{Left} or \textbf{Right}, a \textit{gap} requirement is added to the corresponding \texttt{SUBCAT} variable. This allows gap percolation to be probabilistically modeled; it “percolates” up the parse tree, and constituency decisions are conditioned on its presence in the same way they would be on other aspects of the history.

Collins reports the results in table 2.1. All three of his models outperform the previous state of the art, and each of the aforementioned modifications leads to an improvement in performance.

### 2.5.3 Attribute-value grammars and maximum entropy-based parsing

As mentioned previously, stochastic AV grammars differ from context-free varieties in the way probabilities may be assigned to trees. It is not sufficient to consider the probability of a tree as equal to the joint probabilities of all of its component rules, since the probability of the expansion of one rule in one part of the tree may be influenced by the expansion of another rule elsewhere. Such cases of influence are known as statistical dependencies, and can easily thwart attempts at modeling if their existence is overlooked. Since the grammars dealt with in this thesis are all AV grammars of some variety, it is important that the approach taken be one which is robust to handling statistical dependencies.

In chapter 4, I describe the problem of stochastic modeling of AV grammars. Log-linear models of the variety described in chapter 3 will play a major part in this task, since such models allow statistical dependencies to be accurately modeled. In chapter 4 a number of different approaches to parsing and grammar
modeling in this framework will be described, then in chapter 5 I will introduce the approach used in the present thesis.

2.5.4 Other approaches

Ratnaparkhi (1997) employs a maxent approach to history-based parsing, along the lines of SPATTER, but employing a log-linear classifier to decide on what course to take as it progresses. This approach will be described briefly in chapter 4, along with other uses of log-linear or maxent techniques in grammar disambiguation.

A related approach to parsing the Wall Street Journal corpus is taken in Charniak (1999). This parser works similarly to that of Collins (1997), but adds still further information, such as marked coordinate structures and standard interpolation of the probabilities with backoff conditioning factors. Also, this parser makes use of higher order Markov grammars, with its best performance using a 3rd order Markov grammar. In addition to these modifications, Charniak introduces a so-called “maximum-entropy inspired” aspect of the parser, which appears to be essentially a method of hand-modifying conditional probabilities to more accurately approximate intuitive dependencies and independencies. Charniak’s results suggest that this is useful; he reports superior results to any of the preceding parsers on the same data, as seen in table 2.1.

The approach described in Charniak (1999) does not actually employ the maximum entropy technique, but is merely inspired by maxent. It is suggested that the resulting conditional probabilities may be used as starting points for maximum entropy parameter setting to speed up convergence to optimal values.

As of this writing, the best results on this task, reported by Henderson and Brill (1999), have been reached using an approach of combining the state-of-the-art parsers of Collins (1997), Charniak (1997), and Ratnaparkhi (1997). Henderson and Brill (1999) compare several approaches to combining the parsers to yield performance results superior to any of the parsers working individually. The best results reported employ a naive Bayes classification scheme to determine which constituents from each of the parsers to include in the final parse. Table 2.2 shows the reported f-score results for the best individual parser, a “constituent voting” scheme whereby constituents are included in the generated parse based on whether enough of the parsers agree on them, and the Bayes classifier.

2.6 Commonalities in statistical approaches

All of the approaches which have been mentioned thus far have a number of factors in common, which have been repeatedly demonstrated to be useful to the task of disambiguation. All consider statistics relevant to assigning probabilities to individual rule expansions, whether by counting expansions explicitly in the
2.6. COMMONALITIES IN STATISTICAL APPROACHES

<table>
<thead>
<tr>
<th>System</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Individual Parser</td>
<td>89.67</td>
</tr>
<tr>
<td>Constituent Voting</td>
<td>90.78</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>91.25</td>
</tr>
</tbody>
</table>

Table 2.2: Results for state-of-the-art parsers used in combination.

corpus and creating a treebank grammar, or by using statistics derived from partial expansions to assign probabilities to arbitrary expansions by use of a Markov grammar. Each incorporates head information and lexical and POS tag information for contexts local to the decisions being made. The modifications made which improved the parsers suggest that information about the syntactic category of constituents, lexical heads, subcategorization frames, and syntactic gaps is useful as well as other more specific information about sister relations. In chapter 5 these various considerations will be incorporated into an approach to building simple yet informative compositional features for maximum entropy models.

It is important also to note the differences between the background work discussed in this chapter and the research reported on in chapters 6 and 7. All three of the approaches compared in the preceding sections seek to apply machine learning techniques to learning both grammar rules and their probabilities from a corpus. In the present work, a known grammar is assumed, and only the probabilities of parses is learned. The salient structural considerations are analogous, but the task is rather different and involves a different approach. Background research which is more along the lines of the work in this thesis is presented in chapter 4.