# 2

# Natural Language Parsing

Research on natural language parsing has over a period of several decades produced a wealth of knowledge concerning different methods for automatic syntactic analysis. Most of the results, however, concern formal grammars and algorithms that are only indirectly related to the more practical problem of analyzing syntactic structure in naturally occurring texts. This has led to a somewhat paradoxical situation where, despite the increase in knowledge about the complexity of problems and algorithms for formal grammars, we know relatively little about the formal properties of text parsing. In fact, it is still not clear that there is a well-defined parsing problem for natural language text that is computable in the strict sense.

In this chapter, we will begin by contrasting the two notions of parsing, the well-defined parsing problem for formal grammars, familiar from both computer science and computational linguistics, and the more open-ended problem of parsing unrestricted text in natural language, which is the focus of the investigations in this book. We will then review different strategies for text parsing, including both grammar-driven and data-driven approaches, and discuss the different kinds of problems that arise with different methods. On the basis of this discussion, we will then define the basic requirements of robustness, disambiguation, accuracy and efficiency, which are central to the investigations of text parsing in this book, and discuss evaluation criteria for each of the requirements.

The primary goal of this chapter is to set the scene for the exploration of inductive dependency parsing in later chapters, by defining the basic problems and evaluation criteria, but in doing so we will also have reason to review some of the more important trends in recent research on natural language parsing. First of all, however, we need to say a few words about the desired output of the parsing process, i.e., about syntactic representations for natural language sentences.

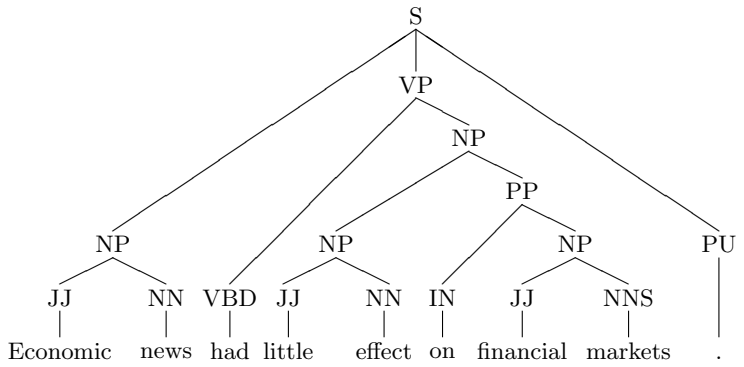


**Fig. 2.1.** Constituent structure for English sentence from the Penn Treebank

## 2.1 Syntactic Representations

The type of syntactic representation that has been dominant during the last fifty years, both in theoretical linguistics and in natural language processing, is based on the notion of *constituency*. In this representation, a sentence is recursively decomposed into smaller segments, called *constituents* or *phrases*, which are typically categorized according to their internal structure into *noun phrases*, *verb phrases*, etc. Constituency analysis comes from the structuralist tradition represented by Bloomfield (1933) and was formalized in the 1950s in the model of phrase structure grammar, or context-free grammar (Chomsky, 1956). Figure 2.1 shows a typical constituency representation of an English sentence, taken from the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993, 1994).[1]

A wide range of different theories about natural language syntax are based on constituency representations. In addition to the theoretical tradition of Chomsky (1957, 1965, 1981, 1995), this includes frameworks that are prominent in computational linguistics, such as Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2000), Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), Tree Adjoining Grammar (TAG) (Joshi, 1985, 1997), and Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1987, 1994).

Another type of syntactic representation, which has a long tradition in descriptive linguistics especially in Europe, is instead based on the notion of *dependency*. In this representation, a sentence is analyzed by connecting its words by binary asymmetrical relations, called *dependencies*, which are

[1] The representation is equivalent to the treebank annotation except that the part-of-speech category '.' has been replaced by PU (for *punctuation*) to avoid a name clash with the terminal '.'. This will simplify exposition later on.

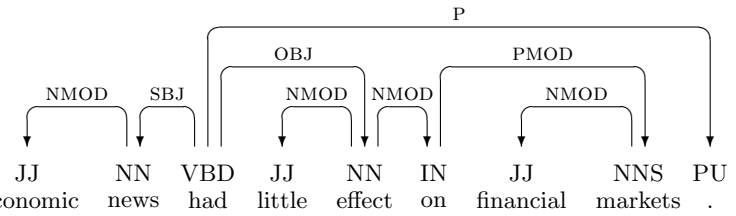


**Fig. 2.2.** Dependency structure for English sentence (same as figure 1.1)

typically categorized according to their functional role into *subject*, *object*, etc. Figure 1.1, repeated here for convenience as figure 2.2, shows a typical dependency representation of the same sentence as in figure 2.1.

According to some scholars, dependency analysis can be traced back to Antiquity (Kruijff, 2002), but the start of the modern tradition is usually taken to be the work of Tesnière (1959). Linguistic theories that are based on dependency representations include Word Grammar (Hudson, 1984, 1990), Functional Generative Description (Sgall et al., 1986), Lexicase (Starosta, 1988), and Meaning-Text Theory (Mel'čuk, 1988).

A third kind of syntactic representation is found in *categorial grammar*, which connects syntactic (and semantic) analysis to inference in a logical calculus. The syntactic representations used in categorial grammar are essentially proof trees, which cannot be reduced to constituency or dependency representations, although they have affinities with both. The categorial grammar tradition goes back to Ajdukiewicz (1935), was taken up in the 1950s by Bar-Hillel (1953) and Lambek (1958), and was used by Montague (1970, 1973) in his influential work on model-theoretic semantics. More recent frameworks that are prominent in the literature on syntactic parsing includes Combinatory Categorial Grammar (CCG) (Steedman, 2000), Type-Logical Grammar (Morrill, 1994, 2000), and Grammatical Framework (Ranta, 2004; Ljunglöf, 2004).

One final type of representation that is widely used in modern syntactic theories is the notion of a *feature structure*, or *attribute-value* representation (Johnson, 1988; Carpenter, 1992), which is usually formalized as a directed acyclic graph. Theories that make use of feature structures are often said to be *unification-based* (Shieber, 1986), since *unification* is the major operation used to combine information from different structures. Feature structures are often combined with constituency representations, either by decorating tree nodes with feature structures, as in GPSG (Gazdar et al., 1985), or by adding new layers of representation over and above the constituency representation, as in LFG (Kaplan and Bresnan, 1982), or by using feature structure representations to encode all aspects of linguistic structure, including constituency,

as in HPSG (Pollard and Sag, 1994). However, feature structures are also found in dependency-based theories, such as Dependency Unification Grammar (DUG) (Hellwig, 2003), and in categorial grammar frameworks, such as Categorial Unification Grammar (CUG) (Uszkoreit, 1986) and Unification Categorial Grammar (UCG) (Zeevat et al., 1991).

Throughout this book, we will mainly be concerned with dependency representations, which will be discussed in more depth in chapter 3. However, in this chapter we will try to abstract away from the particular representations used and concentrate on issues in natural language parsing that cut across different frameworks. Thus, when we speak about syntactic parsing as the problem of assigning an *analysis* to an input string, it will be understood that the analysis is a syntactic representation as defined by the relevant framework. To some extent, this means that we will be comparing apples and oranges, since the problems involved in parsing are not independent of the nature of syntactic representations. Still, we feel that different frameworks have enough in common to make a general discussion fruitful, although we will also make reference to different syntactic representations when this is relevant.

## 2.2 Two Notions of Parsing

The term *parsing*, derived from the Latin *pars orationis* (parts of speech), was originally used to denote the grammatical explication of sentences, as practiced in elementary schools. The term was then borrowed by linguistics and computer science, where it has acquired a specialized sense in connection with the theory of formal languages and grammars. However, in practical applications of natural language processing, the term is also used to denote the syntactic analysis of sentences in text, without reference to any particular formal grammar, a sense which is in many ways quite close to the original grammar school sense.

In other words, there are at least two distinct notions of parsing that can be found in the current literature on natural language processing, notions that are not always clearly distinguished. Although we are certainly not the first to notice this ambiguity, we feel that it has not been given the attention that it deserves. While it is true that there are intimate connections between the two notions, they are nevertheless independent notions with quite different properties in some respects. In order to highlight these differences, we will now proceed to a contrastive examination of the notions of *grammar parsing* and *text parsing*.[2]

[2] The term *text* in *text parsing* is not meant to exclude spoken language, but rather to emphasize the relation to naturally occurring language use. Although we will have nothing to say about the parsing of spoken utterances in this book, we want the notion of text parsing to encompass both written texts and spoken dialogues. An alternative term would be *discourse parsing*, but this would give rise to misleading associations of a different kind.

| | |
|---|---|
| S → NP VP PU | JJ → Economic |
| VP → VP PP | JJ → little |
| VP → VBD NP | JJ → financial |
| NP → NP PP | NN → news |
| NP → JJ NN | NN → effect |
| NP → JJ NNS | NNS → markets |
| PP → IN NP | VBD → had |
| PU → . | IN → on |

**Fig. 2.3.** Context-free grammar for a fragment of English

### 2.2.1 Grammar Parsing

The notion of grammar parsing is intimately tied to the notion of a formal grammar $G$ defining a formal language $L(G)$ over some (terminal) alphabet $\Sigma$. Formally, an alphabet $\Sigma$ is a set of elementary symbols, and a formal language over $\Sigma$ is a subset of the set $\Sigma^*$ of all strings formed from symbols in $\Sigma$. A grammar $G$ is a formal system for deriving strings over some alphabet $\Sigma$, and the language $L(G)$ defined by $G$ is the set of all strings $x$ derivable in $G$. Moreover, each (canonical) derivation of a string $x$ corresponds to a syntactic analysis of $x$ according to $G$. The *parsing problem* can then be defined as follows:

> Given a grammar $G$ and an input string $x \in \Sigma^*$, derive some or all of the analyses assigned to $x$ by $G$.

This is sometimes called the *universal* parsing problem for grammars of a certain type. It is also possible to define the parsing problem relative to a *particular* grammar $G$ of some type, although this is usually considered less interesting (Barton et al., 1987). The analysis of formal grammars and their parsing problems goes back to the pioneering work of Noam Chomsky and others in the 1950s and continues to be a very active area of research. A classic introduction to the field is Hopcroft and Ullman (1979), which recently appeared in a new and revised edition (Hopcroft et al., 2001).

The most widely used type of formal grammar, in computer science as well as computational linguistics, is the *context-free grammar* (CFG) of Chomsky (1956), which is equivalent to the independently defined *Backus-Naur Form* (BNF) (Backus, 1959). Figure 2.3 shows a context-free grammar defining a fragment of English including the sentence analyzed in figure 2.1. One of the analyses assigned to this sentence by the grammar is the *parse tree* depicted in figure 2.1, which corresponds to a canonical derivation of the word string.[3]

[3] Without the restriction to some canonical form of derivation (e.g., a leftmost or a rightmost derivation), there is generally more than one derivation of the same parse tree.

Over the years, a variety of formal grammars have been introduced, many of which are more expressive than context-free grammar and motivated by the desire to provide a more adequate analysis of natural language syntax. This development started with the transformational grammars of Chomsky (1957, 1965) and has continued with many of the theoretical frameworks mentioned in the previous section, such as LFG and HPSG. In recent years, there has been a special interest in so-called mildly context-sensitive grammars (Joshi, 1985), exemplified by TAG and CCG, which appear to strike a good balance between linguistic adequacy and computational complexity.

Solving the universal parsing problem for a particular type of grammar requires a *parsing algorithm*, i.e., an algorithm that computes analyses for a string $x$ relative to a grammar $G$. Throughout the years a number of parsing algorithms for different classes of grammars have been proposed and analyzed. For context-free grammar, some of the more well-known algorithms are the Cocke-Kasami-Younger (CKY) algorithm (Kasami, 1965; Younger, 1967), Earley's algorithm (Earley, 1970), and the left corner algorithm (Rosenkrantz and Lewis, 1970). These algorithms all make use of tabulation to store partial results, which potentially allows exponential reductions of the search space and thereby provides a way of coping with ambiguity. In the computational linguistics literature, this technique is best known as *chart parsing* (Kay, 1980; Thompson, 1981). This type of method, which constitutes a form of *dynamic programming* (Cormen et al., 1990), can also be generalized to more expressive grammar formalisms.

Deterministic parsing algorithms, such as LR parsing (Knuth, 1965), can only handle restricted subsets of context-free grammars and have their main use in compilers for programming languages. However, deterministic parsing techniques based on shift-reduce parsing (Aho et al., 1986) have also been applied to natural language parsing, often with the ambition to model human sentence processing (Marcus, 1980; Shieber, 1983). In addition, the generalized LR (GLR) parsing algorithm proposed by Tomita (1987) can handle arbitrary context-free grammars and avoids exponential search by using a graph-structured stack and tabulation of partial results.

Traditional methods for parsing can be described as *constructive* in the sense that they analyze sentences by constructing syntactic representations in accordance with the rules of a given grammar. An alternative is to use an *eliminative* parsing method, which treats the grammar as a set of constraints and views parsing as a constraint satisfaction problem, which can be solved by successively eliminating analyses that violate constraints until only valid analyses remain. This strategy presupposes a compact representation of the space of possible analyses and has therefore mainly been used with syntactic representations that are reducible to an assignment of categories or structural attachments to word tokens, as in Constraint Grammar (CG) (Karlsson, 1990; Karlsson et al., 1995), Constraint Dependency Grammar (CDG) (Maruyama, 1990; Harper and Helzerman, 1995; Menzel and Schröder, 1998), and Topological Dependency Grammar (TDG) (Duchier, 1999). In special cases, this

parsing strategy can also be implemented using finite state techniques, as in Parallel Constraint Grammar (Koskenniemi, 1990, 1997). Besides constructive and eliminative parsing methods, we may also distinguish a *transformational* approach, where parsing starts from some kind of default representation and applies grammatical rules that transform this to an output representation (Brill, 1993; Foth et al., 2004).

We will make no attempt to review the vast literature on grammar parsing here but will limit ourselves to a few observations concerning the properties of the parsing problem and the methods used to solve it.[4] First of all, it is worth noting that the parsing problem for a class of grammars is a well-defined *abstract problem* in the sense of algorithm theory (Cormen et al., 1990), i.e., a relation between a set $I$ of inputs, which in this case are pairs consisting of a grammar $G$ and a string $x$, and a set $O$ of outputs, which are syntactic representations of strings in $L(G)$. A parsing algorithm provides a solution to this problem by computing the mapping from arbitrary inputs to outputs.

Secondly, the parsing problem for formal grammars is intimately tied to the corresponding *recognition problem*, i.e., the problem of deciding whether the string $x$ is in $L(G)$. It is only strings in $L(G)$ that receive an analysis in the parsing process, and most parsing algorithms in fact solve the recognition problem simultaneously.

Thirdly, we note that the analyses to be assigned to a particular input string $x$ are completely defined by the grammar $G$ itself. For example, if $G$ is a context-free grammar, we may be interested in the set of distinct parse trees that result from derivations of $x$ from the start symbol $S$ of $G$. In principle, this means that the correctness of a parsing algorithm can be established without considering any particular input strings, since the set of all input-output pairs are given implicitly by the grammar $G$ itself.

The abstract nature of the grammar parsing problem is reflected in the evaluation criteria that are usually applied to parsing methods in this context. For example, a parsing algorithm is said to be *consistent* if, for any grammar $G$ and input string $x$, it only derives analyses for $x$ that are licensed by $G$; it is said to be *complete* if, for any $G$ and $x$, it derives *all* analyses for $x$ that are licensed by $G$. For example, the grammar in figure 2.3 is ambiguous and assigns to our example sentence not only the analysis in figure 2.1 but also the analysis in figure 2.4. Thus, a complete parsing algorithm must compute both these analyses, while a consistent algorithm must not compute any other analysis. However, both consistency and completeness of an algorithm can be proven without considering any particular grammar $G$ or input string $x$, given the formal definition of the class of grammars and the relevant notions of derivation and representation.

The same goes for considerations of efficiency, where proofs of complexity, either for particular parsing algorithms or for classes of grammars, provide the

[4] For general overviews of grammar parsing techniques, with special reference to natural language parsing, see Samuelsson and Wirén (2000) and Carroll (2003).

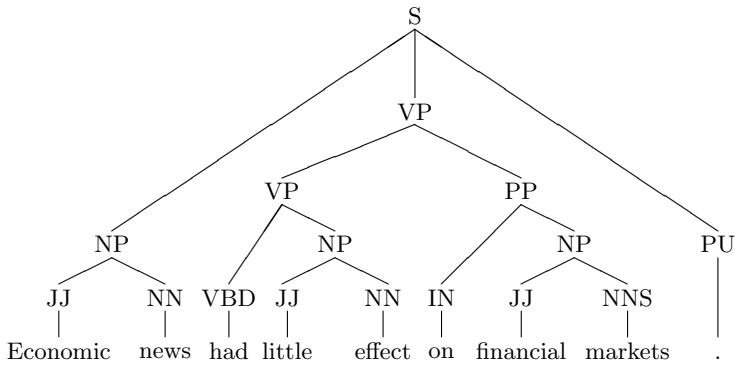


**Fig. 2.4.** Alternative constituent structure for English sentence (cf. figure 2.1)

most relevant tools for evaluation. Research on the complexity of linguistically motivated classes of grammars was pioneered by Barton et al. (1987) and has been followed by a large number of subsequent studies. For a context-free grammar $G$, parsing can be performed in $O(n^3)$ time, where $n$ is the length of the input string $x$, using one of the dynamic programming algorithms mentioned earlier. For mildly context-sensitive grammars, parsing complexity is still polynomial — typically $O(n^6)$ — while for more expressive formalisms running time becomes exponential in the worst case.

Although complexity results often need to be supplemented by practical running time experiments, as shown for example by Carroll (1994), the role of empirical evaluation is rather limited in grammar parsing, especially as far as correctness is concerned. This follows from the fact that grammar parsing, as defined here, is an abstract and mathematically well-defined problem, which can be studied using formal methods only. One setting where such studies have been performed very fruitfully is within the framework known as *deductive parsing* (Shieber et al., 1995).

### 2.2.2 Text Parsing

The notion of text parsing applies to concrete manifestations of a language $L$, where we do not necessarily assume that $L$ is a formal language. In particular, we are interested in the case where $L$ is a natural language, or possibly a restricted subset of a natural language. We assume that a *text* in a language $L$ is a sequence $T = (x_1, \ldots, x_n)$ of sentences $x_i$, and we characterize the *text parsing* problem as follows:

> Given a text $T = (x_1, \ldots, x_n)$ in the language $L$, derive the correct analysis for every sentence $x_i \in T$.

The term *sentence* should be understood here in the sense of *text sentence* rather than *system sentence* (Lyons, 1977), i.e., it refers to a segment of text without any specific assumptions about syntactic completeness or other structural properties. What constitutes a sentence in this sense may differ from one language to the other and may not always be completely clear-cut. For the time being we will simply disregard this problem, although it is well-known that the problem of sentence segmentation in text processing is far from trivial (Palmer, 2000). It is also worth pointing out that the concept of a text as a sequence of sentences is an abstraction in the sense that it ignores all aspects of non-sequential structure that resides, e.g., in the graphical layout of printed texts. This seems like an appropriate abstraction for written text, which is what we are concerned with in this study, but we will probably need a different abstraction for spoken dialogue, where different principles of structural organization are at play and where the notion of sentence has a more unclear status.

To exemplify the notion of text parsing, let us return again to the example sentence from figure 2.1. In its original context, which is a text taken from the Wall Street Journal and included in the Penn Treebank, this sentence has an interpretation that corresponds to the analysis in figure 2.1 — rather than the alternative analysis in figure 2.4. Therefore, the former analysis is the one and only correct analysis in the context of text parsing.

Let us now return to the observations made concerning grammar parsing in the previous section and see in what respects text parsing is different. First of all, it is not clear that text parsing is a well-defined abstract problem in the same sense as grammar parsing, especially not when we consider texts in a natural language. It is true that text parsing has the structure of a mapping problem, but in the absence of a formal definition for the language $L$, there is no precise delimitation of the input set. Moreover, even if we can agree on the formal properties of output representations, there is no formal grammar defining the correct mapping from inputs to outputs. For example, the syntactic representation in figure 2.1 is clearly of the kind that can be defined by a context-free grammar. But according to our conception of the text parsing problem, there is no specific instance of this formal grammar that defines the mapping from an input string to a specific representation. In other words, despite a superficial similarity to the definition of grammar parsing, our characterization of the text parsing problem is not a formal definition at all.

One way of looking at the problem is instead to say that it is an empirical *approximation problem*, where we try to approximate the correct mapping given increasingly large but finite samples of the mapping relation. Needless to say, this is a view that fits very well with the data-driven approach to text parsing, which will be discussed in a later section. However, the main point right now is simply that, unlike grammar parsing, the problem of text parsing lacks a precise definition in formal terms.

Secondly, text parsing lacks the connection between parsing and recognition that we observed for grammar parsing. This is a direct consequence of the fact that the input language is not formally defined, which means that recognition is not a well-defined problem. Therefore, we can no longer *require* that an input string be part of the language to be analyzed. In most cases, we instead have to *assume* that any text sentence is a valid input string. And if we want to be able to reject some input strings as ill-formed, then we cannot refer to a formal language definition but must appeal to some other criterion.

Thirdly, while there is no reference to a grammar in the definition of text parsing, there is instead a reference to the sequence of sentences that provide the textual context for each sentence to be analyzed. This is based on the assumption that text parsing deals with language use, and that the analysis assigned to a sentence is sensitive to the context in which it occurs. In particular, we assume that each text sentence has a single correct analysis, even if the string of words realizing the sentence may be found with other interpretations in other contexts. In other words, text parsing entails disambiguation.

However, the absence of a formal grammar also means that we need some external criterion for deciding what is the correct analysis for a given sentence in context. For natural languages, the obvious criterion to use is human performance, meaning that an analysis is correct if it coincides with the interpretation of competent users of the language in question. This leads to the notion of an *empirical gold standard*, i.e., a reference corpus of texts, where each relevant text segment has been assigned its correct analysis by a human expert. In the case of syntactic parsing, the relevant segments are sentences and the corpus will normally be a *treebank* (Abeillé, 2003b; Nivre, forthcoming). Thus, our reason for saying that the analysis given in figure 2.1 is correct is simply that this is the analysis found in the Penn Treebank.

The use of treebank data to establish a gold standard for text parsing is problematic in many ways, and we will return to this problem in chapter 5. For the time being, we will simply note that even if a gold standard treebank can be established, it will only provide us with a finite sample of input-output pairs, which means that any generalization to an infinite language will have to rely on statistical inference. This is in marked contrast to the case of grammar parsing, where the consistency and completeness of parsing algorithms, for any grammar and any input, can be established using formal proofs.

The empirical nature of the text parsing problem is reflected also in the evaluation criteria that are applied to parsing methods in this context. Since notions of consistency and completeness are meaningless in the absence of a formal grammar, the central evaluation criterion is instead the empirical notion of *accuracy*, which is standardly operationalized as agreement with gold standard data and which will be discussed in detail later on. However, it is important to remember that, even though it is often difficult to apply formal methods to the text parsing problem itself given its open-ended nature, the parsing methods we develop to deal with this problem can of course be subjected to the same rigorous analysis as algorithms for grammar parsing. Thus,

if we are interested in the efficiency of different methods, we may use results about theoretical complexity of algorithms as well as empirical running time experiments. Formal methods can also be used to study aspects of robustness and disambiguation, as we shall see later on. However, for the central notion of accuracy, there seems to be no alternative but to rely on empirical evaluation methods, at least not given the current state of our knowledge.

### 2.2.3 Competence and Performance

The discussion of grammar parsing and text parsing leads naturally to a consideration of the well-known distinction between *competence* and *performance* in linguistic theory (Chomsky, 1965).[5] It may be tempting to assume that grammar parsing belongs to the realm of competence, while text parsing is concerned with performance. After all, the whole tradition of generative grammar in linguistics is built on the idea of using formal grammars to model linguistic competence, starting with Chomsky (1957, 1965). The idea that natural languages can be modeled as formal languages unites theorists as different as Chomsky and Montague (1970), although it is much less prominent in recent formulations of Chomsky's theory (Chomsky, 1981, 1995). Within this tradition, it might be natural to view the study of grammar parsing, when applied to natural language, as the study of idealized human sentence processing.

The traditional notion of linguistic competence has recently been called into question, and it has been suggested that many of the properties typically associated with linguistic performance, such as frequency effects and probabilistic category structure, also belong to our linguistic competence (Bod, Hay and Jannedy, 2003). We will not pursue these complex and controversial issues here, and the nature of linguistic competence, fascinating as it is, falls outside the scope of this study.

On the other hand, it seems quite clear that text parsing is concerned with linguistic performance, at least if we want to use text parsing methods to build practical systems that can handle naturally occurring texts. This means that a model of linguistic competence is of use to us only if it can be coupled with an appropriate model of performance. So, regardless of whether grammar parsing is a good model of linguistic competence or not, it is still an open question what role it has to play in text parsing (cf. Jensen, 1988; Chanod, 2001).

This question will be discussed in detail in the next section, but before we leave the topic of grammar parsing as such, it should be emphasized that this is in any case a very fruitful area of research. Investigations of the complexity of formal grammars and their parsing algorithms have applications in many

[5] Before Chomsky, similar distinctions had been proposed by De Saussure (1916), between *langue* and *parole*, and by Hjelmslev (1943), between *system* and *process*, among others.

areas of computer science, and work on the complexity of linguistically motivated formalisms has had a profound influence on the development of natural language parsing, including the approach investigated in this book.

## 2.3 Methods for Text Parsing

The main conclusion from the preceding section is that grammar parsing and text parsing are in many ways radically different problems and therefore require different methods. In particular, grammar parsing is an abstract problem, which can be studied using formal methods and internal evaluation criteria, while text parsing is an empirical problem, where formal methods need to be combined with experimental methods and external evaluation criteria. In this section, we will go on to discuss methods that have been proposed for text parsing, which is the problem that concerns us here. Some of these methods crucially involve grammar parsing; others do not.

Following Carroll (2000), we distinguish two broad types of strategy: the *grammar-driven approach* and the *data-driven approach*. However, we want to emphasize at the outset that these types are in a sense stereotypes, representing extreme strategies, which means that many existing approaches actually combine elements of both. In fact, at the end of this section we will even go so far as to suggest that recent developments in the field can be seen as signs of convergence between these apparently opposite strategies. Nevertheless, we believe that it is instructive to start out by considering them as alternative methods, since this will highlight the way they tackle the different problems that arise in parsing unrestricted natural language text.

By necessity, any method for text parsing must rely on an approximation, where a well-defined abstract problem is used as a model of the real practical text parsing problem. By computing a solution to the abstract model problem, we can approximate a solution to the real problem. But the adequacy of this approximation can only be assessed through empirical evaluation. The main difference between the grammar-driven and the data-driven approach lies in the type of abstract problem that is chosen to model the text parsing problem.

### 2.3.1 Grammar-Driven Text Parsing

In the grammar-driven approach, text parsing is modeled by the abstract problem of grammar parsing. Hence, a formal grammar $G$ is used to define the language $L(G)$ that can be parsed and the class of analyses to be returned for each string in the language. A grammar parsing algorithm is then used to compute the analyses of a given input string, as described in section 2.2.1. The grammar may be hand-crafted or it may be wholly or partially induced from corpus data. It may be coupled with a statistical model for parse selection, and it may allow partial analyses to achieve robustness. But the essence of

the grammar-driven approach, as understood here, is that it is based on a grammar $G$ defining a formal language $L(G)$.

Given our characterization of text parsing in section 2.2.2, it is clear that the grammar-driven approach is based on a crucial assumption, namely that the formal language $L(G)$ is a reasonable approximation of the language $L$ that we want to process. In practice, it is arguably the case that most if not all of the formal grammars that have been developed for natural languages to date fail to meet this assumption, and the formal language $L(G)$ is at best a restricted subset of the natural language $L$. This does not undermine the grammar-driven approach in principle, since it is always possible to argue that advances in linguistic theory will lead to successively better approximations, but it does create important problems for practical applications of grammar-driven text parsing in the meantime. Many of the research directions in natural language parsing during the last two decades can be seen as motivated by the desire to overcome these problems.

An analogy with syntactic parsing in compilers for programming languages may be illuminating at this point. In a way, parsing computer programs is also a kind of text parsing, but there is a crucial difference in that we know from the start that the approximation $L(G) = L$ can be made perfect, simply because the programming language $L$ is itself a formal language with a precise syntax definition. Moreover, this definition is usually expressible in a carefully restricted subclass of context-free grammar. This does not necessarily mean that compilers always use grammar-driven parsers in practice, or that they use a grammar $G$ such that $L(G)$ exactly coincides with $L$, but it means that it is always possible to find out whether the approximation is perfect or not. For natural languages, this is unfortunately not the case.

One of the hardest problems for the grammar-driven approach has traditionally been to achieve *robustness*, where robustness can be defined as the capacity of a system to analyze any input sentence. Alternatively, we may view robustness as a matter of degree and say that a system is more robust if it can adequately handle a larger proportion of the input data. In either case, the shortcomings of grammar-driven systems in this respect can be traced back to the fact that some input sentences $x_i$ in a text $T$ are not in the language $L(G)$ defined by the formal grammar $G$.

Theoretically speaking, it is possible to distinguish two problematic cases where $x_i \notin L(G)$. In the first case, $x_i$ is a perfectly well-formed sentence of the language $L$ and should therefore also be in $L(G)$ but is not. This is sometimes referred to as the problem of *coverage*, since it should be eliminated by increasing the coverage of the grammar. In the second case, $x_i$ is considered not to be part of $L$, and should therefore not be in $L(G)$ either, but nevertheless has a reasonable syntactic analysis. This can then be called the problem of robustness proper. Examples of the latter type include sentences where some word is misspelled or even omitted, but where it is nevertheless possible to analyze the syntactic structure of (the rest of) the sentence.

For example, if our example sentence from figure 2.1 had contained the token *impact* or the token *effact*, instead of the token *effect*, then the sentence would not have been included in the language defined by the grammar in figure 2.3. In the first case, this would be a problem of coverage, since *impact* is a real English word, which can occur in the same structural position as *effect*. In the second case, it would be a problem of robustness, since *effact* is not a word of English.

However, even though there are many clear-cut examples like these, there are also many cases where it is difficult to decide whether a sentence that is not in $L(G)$ is in $L$, at least without making appeal to a prescriptive grammar for the natural language $L$. For certain practical applications, such as grammar checking, it is obviously both relevant and necessary to use this kind of information, but it can be problematic in the general case. Moreover, since it is difficult to apply the distinction between coverage and robustness to approaches that are not grammar-driven, we will not try to maintain this distinction in general but simply treat all failures to analyze input sentences as problems of robustness.[6]

As pointed out by Samuelsson and Wirén (2000), there are essentially two methods that have been proposed to overcome the robustness problem for grammar-driven systems. The first is to relax the grammatical constraints of $G$ in such a way that a sentence outside $L(G)$ can be assigned a complete analysis (Jensen and Heidorn, 1983; Mellish, 1989). This method has the potential drawback that the number of relaxation alternatives that are compatible with analyses of the complete input may become extremely large, and this in turn will aggravate the problem of disambiguation to be discussed below.

The second method is to maintain the constraints of $G$ but to recover as much structure as possible from well-formed fragments of the sentence. This leads to the notion of *partial parsing*, which has been explored within many different frameworks such as deterministic parsing (Hindle, 1989, 1994), chart parsing (Lang, 1988), finite state parsing (Ejerhed, 1983; Koskenniemi, 1990, 1997; Abney, 1991, 1996; Oflazer, 2003), and Constraint Grammar (CG) parsing (Karlsson, 1990; Karlsson et al., 1995), and which nowadays includes not only recovery techniques but also approaches that never attempt to build a complete syntactic structure. Compared to constraint relaxation, partial parsing has the advantage of hiding ambiguity instead of increasing it, since a partial syntactic analysis can be viewed as an underspecified representation of a complete analysis and therefore fails to exhibit some of the ambiguities that distinguish non-equivalent complete analyses. For example, a chunking analysis can be viewed as a partial, underspecified constituency analysis, and a CG analysis as a partial dependency analysis.

In this way, partial parsing can be seen as a way to sacrifice completeness and depth of analysis to improve robustness and efficiency (Abney, 1997).

[6] For a more extensive discussion of robustness in natural language parsing, see Menzel (1995), Junqua and Van Noord (2001) and Basili and Zanzotto (2002).

This may be useful especially in applications that do not require a complete syntactic analysis. However, it is also possible to view partial parsing as a way to break down the complex parsing problem into subproblems that may be easier to manage. This leads to the notion of *cascaded partial parsing* (Abney, 1996), where full parsing is achieved through a sequence of partial parsers, where each parser takes as input the output of the preceding one. A variant of this approach is the use of *supertagging*, pioneered by Bangalore and Joshi (1999), where the words of a sentence are annotated with rich structural or functional categories in order to facilitate the derivation of a syntactic structure in a second step (Joshi and Sarkar, 2003). In a similar vein, the framework of Functional Dependency Grammar (FDG) (Tapanainen and Järvinen, 1997; Järvinen and Tapanainen, 1998) uses CG parsing as a form of supertagging for the construction of dependency structures.

Although we have made a distinction between constraint relaxation and partial parsing as two strategies for achieving robustness in grammar-based text parsing, it should be pointed out that practical implementations often combine the two techniques. For example, in a CG parser that constructs a partial analysis by applying constraints in an eliminative fashion, there is usually a mechanism to prevent the elimination of the last analysis, which is in fact a general mechanism for constraint relaxation that guarantees that the system will output an analysis even if there is no analysis that satisfies all the constraints of the grammar.

Another major problem for grammar-driven text parsing is the problem of *disambiguation*, which is caused by the fact that the number of analyses assigned to a sentence $x_i$ by the grammar $G$ can be very large, while text parsing requires that a small number of analyses (preferably a single one) are selected as appropriate in the context of the text $T$. For example, the grammar in figure 2.3 assigns two different analyses to our example sentence (cf. figure 2.1 and figure 2.4), which means that a text parsing system using this grammar must incorporate a mechanism for selecting one of them as correct in the given context.[7]

Again, we can make a theoretical distinction between two reasons that the grammar parser outputs more than one analysis for a given string. On the one hand, we have cases of true ambiguity, i.e., where $x_i$ admits of more than one syntactic analysis in the language $L$, even though only one of them is appropriate in the textual context, and where the grammar $G$ captures this by assigning several analyses to $x_i$. On the other hand, it may be the case that the grammar $G$ contains rules that license analyses for $x_i$ that are never encountered in $L$. The latter problem is sometimes called the *leakage* problem, in allusion to Sapir's famous statement that '[a]ll grammars leak' (Sapir, 1921, 39), or simply *overgeneration*. Although one might argue that it

[7] For this particular sentence, it is not clear that the syntactic ambiguity makes a difference in meaning. In fact, it could even be argued that *have little effect* is a light verb construction and that the analysis in figure 2.4 is more appropriate.

is only the former problem that relates to disambiguation proper, it is again very difficult in practice to draw a sharp distinction between problems of leakage and problems of disambiguation, and we will therefore use the term *disambiguation* for the process of reducing the number of analyses assigned to a string, whether the analyses should be licensed by an underlying grammar or not.

Early work related to the ambiguity problem used specialized grammars for different domains of text. Even though this will not lead to complete disambiguation in all cases, it can drastically reduce the number of analyses assigned to a given string, compared to broad-coverage domain-independent grammars. A more sophisticated variant of this approach, which also allows grammar resources to be reused across domains, is to use machine learning techniques to specialize a grammar to a new domain (Grishman et al., 1984; Samuelsson and Rayner, 1991).

A different approach to disambiguation in grammar-driven text parsing is to use deterministic processing and try to ensure that, as far as possible, a correct decision is made at each nondeterministic choice point corresponding to an ambiguity (Marcus, 1980; Shieber, 1983). As mentioned earlier, this line of research has often been motivated by a desire to model human sentence processing, which is assumed to use deterministic disambiguation in combination with backtracking if necessary. An early version of the framework investigated in this book used a simple grammar together with hand-crafted heuristics to achieve deterministic dependency parsing (Nivre, 2003).

Disambiguation is not independent of the basic parsing methodology, and it could be argued that eliminative and transformational methods are more geared towards disambiguation than traditional constructive methods. In an eliminative framework, parsing and disambiguation can be said to coincide, since parsing is performed by successively eliminating candidate analyses, as in the CG framework or its descendant FDG. On the other hand, this kind of eliminative parsing presupposes that we construct a compact representation of the space of possible analyses, which means that it can also be viewed as a very simple and efficient form of constructive parsing followed by eliminative disambiguation. In the transformational approach, parsing is instead the successive transformation of a single representation, which means that there is only one analysis available at any point in time. This technique has been used for disambiguation in transformation-based parsing (Brill, 1993) and Weighted Constraint Dependency Grammar (WCDG) (Foth et al., 2004).

However, the most common approach to disambiguation in recent years has been the use of statistical information about the text language $L$ to rank multiple competing analyses ($n$-best parsing) or to select a single preferred analysis. There are several ways in which statistical information can be integrated into the grammar-driven approach, but the most straightforward approach is to use a stochastic extension of a formal grammar, the most well-known example being *probabilistic context-free grammar* (PCFG). In a PCFG, every context-free production is associated with a probability $p$ in such a way

| | | | |
|---|---|---|---|
| S → NP VP PU | 1.0 | JJ → Economic | 0.3 |
| VP → VP PP | 0.3 | JJ → little | 0.5 |
| VP → VBD NP | 0.7 | JJ → financial | 0.2 |
| NP → NP PP | 0.2 | NN → news | 0.4 |
| NP → JJ NN | 0.5 | NN → effect | 0.6 |
| NP → JJ NNS | 0.3 | NNS → markets | 1.0 |
| PP → IN NP | 1.0 | VBD → had | 1.0 |
| PU → . | 1.0 | IN → on | 1.0 |

**Fig. 2.5.** Probabilistic context-free grammar for a fragment of English

that the probabilities of all productions with the same left-hand side sum to 1. Thus, the probability of a production $A \to \omega$ is the conditional probability $P(\omega \mid A)$ of the right-hand side $\omega$ given the left-hand side $A$. The probability of a parse tree is then the product of probabilities of all the productions used to construct it (Booth and Thompson, 1973), which amounts to assuming that all production probabilities are mutually independent.

Figure 2.5 shows a PCFG based on the CFG in figure 2.3. Although the actual probabilities assigned to the different rules are completely unrealistic because of the very limited coverage of the grammar, it nevertheless serves to illustrate the notion of a probabilistic grammar. According to this grammar, the probability of the parse tree in figure 2.1 is 0.0000756, while the probability of the parse tree in figure 2.4 is 0.0001134. In other words, using this PCFG for disambiguation, we would prefer the second analysis, which attaches the PP *on financial markets* to the verb *had*, rather than to the noun *effect*. According to the annotation in the Penn Treebank, this would not be the correct choice.

Early work on PCFG parsing used unsupervised machine learning, in particular the Inside-Outside algorithm (Baker, 1979), applied to text corpora to estimate the probabilistic parameters of hand-crafted context-free grammars (Fusijaki et al., 1989; Pereira and Schabes, 1992). But given a treebank with context-free syntactic representations, it is also possible to extract productions directly from the analyses in the treebank and to use frequency counts to estimate probabilistic parameters in a supervised fashion, which results in a so-called *treebank grammar* (Charniak, 1996).

The probabilistic model associated with PCFGs has turned out to be a rather blunt tool for disambiguation, because its independence assumptions are such that it misses dependencies that seem to be important for correct disambiguation. For example, bilexical dependencies, i.e., dependencies between word pairs, are outside the reach of the basic model. This has led people to explore various kinds of lexicalized stochastic grammars, either based on the PCFG model or on other formal grammars such as Lexical Tree Adjoining Grammar (LTAG) (Schabes et al., 1988), for which stochastic versions have been defined by Schabes (1992) and Resnik (1992). In fact, most of the advances in text parsing during the last decade can be traced to the development

of better statistical methods for disambiguation. This often results in a combination of the grammar-driven and data-driven approaches, and we will postpone our discussion of these methods until the next section, where they will be treated together with other data-driven methods for text parsing.

The problems of robustness and disambiguation cannot be studied in isolation from the problem of *accuracy*. If robustness and disambiguation have traditionally been considered the stumbling blocks for grammar-driven text parsing, it is often assumed that this approach has an advantage with respect to accuracy, since the grammar $G$ is meant to guarantee that the analysis assigned to a sentence $x_i$ in a text $T$ is linguistically adequate. However, even if we disregard the leakage problem, this argument is only tenable as long as we do not require robustness and disambiguation. As we have seen above, robustness may require the analysis of strings that are not in the language $L(G)$ defined by the grammar. And disambiguation normally entails discarding most of the analyses assigned to a string by the grammar. All other things being equal, these requirements will decrease the likelihood that a given string $x_i \in T$ is assigned the contextually correct analysis by the parsing system. This means that we need to consider the joint optimization of robustness, disambiguation and accuracy, even if we can decide to prioritize them differently.

The need for joint optimization also includes the final problem that we will consider, namely *efficiency*, which can be a more or less serious problem for the grammar-driven approach depending on the expressivity and complexity of the formal grammars used. For many linguistically motivated frameworks, such as LFG and GPSG, the parsing problem has been shown to be computationally intractable (Barton et al., 1987).[8] It should be remembered, though, that these results concern the theoretically worst case, which for many frameworks occurs only under very special circumstances. Therefore, exponential time algorithms can often be used in practical parsing systems, possibly in combination with special mechanisms to limit the computational effort when the worst-case exponential behavior is encountered (Kaplan et al., 2004). Another approach, which is less common in practice, is to make use of context-free approximations of the full-fledged grammar (Torisawa et al., 2000).

But even for grammars that allow parsing in polynomial time and space, efficiency can be a problem in practical applications. This is the case for many of the mildly context-sensitive grammar formalisms that have been proposed for natural language syntax, such as Tree Adjoining Grammar (TAG) (Joshi, 1985, 1997) and Combinatory Categorial Grammar (CCG) (Steedman, 2000), where the time complexity for parsing is $O(n^6)$ relative to the length $n$ of the input string. Moreover, the requirements of robustness and disambiguation can easily compromise efficiency. Enforcing robustness by relaxing constraints may lead to a combinatorial explosion in the number of possible analyses,

[8] LFG parsing is NP hard, which means that it is probably not computable in polynomial time; GPSG parsing is EXP-POLY hard, which means that it is certainly not computable in polynomial time (Barton et al., 1987).

and disambiguation may require the enumeration of an exponential number of analyses from the compact tabular representation computed during parsing. In addition, both time and space complexity are normally dependent on the size of the grammar, as well as the degree of lexical ambiguity, factors that may in fact dominate the time and space consumption as grammar size grows with the increased coverage required by robustness. However, important progress has been made in recent years to speed up parsing for highly expressive grammar frameworks, as reported, e.g., in Malouf et al. (2000); Oepen and Carroll (2000); Riezler et al. (2002); Miyao et al. (2003); Kaplan et al. (2004); Clark and Curran (2004); Curran and Clark (2004).

Finally, it is worth noting that the most efficient methods for grammar-driven text parsing are those that are based on deterministic algorithms, whether they apply to context-free grammars (Hindle, 1989, 1994), to finite state models (Ejerhed, 1983; Koskenniemi, 1990, 1997; Abney, 1991, 1996; Roche, 1997; Oflazer, 2003), or to automatically induced finite state approximations of context-free grammars (Pereira and Wright, 1997; Nederhof, 1998, 2000; Mohri and Nederhof, 2001).

### 2.3.2 Data-Driven Text Parsing

In the data-driven approach to text parsing, a formal grammar is no longer a necessary component of the parsing system. Instead, the mapping from input strings to output analyses is defined by an inductive mechanism applying to a text sample $T_t = (x_1, \ldots, x_n)$ from the language $L$ to be analyzed. Hence, the abstract problem used to approximate text parsing in this case is a problem of inductive inference, which may or may not be constrained by a formal grammar. In general, we can distinguish three essential components in a data-driven text parser:

1. A formal model $M$ defining permissible analyses for sentences in $L$.
2. A sample of text $T_t = (x_1, \ldots, x_n)$ from $L$, with or without the correct analyses $A_t = (y_1, \ldots, y_n)$.
3. An inductive inference scheme $I$ defining actual analyses for the sentences of any text $T = (x_1, \ldots, x_n)$ in $L$, relative to $M$ and $T_t$ (and possibly $A_t$).

This may not be the most familiar way to describe data-driven text parsing, but we believe that this characterization provides a useful abstraction over many existing approaches and furthermore allows a fruitful comparison to the grammar-driven approach.[9]

The first thing to note is that the formal model $M$ may in fact be a formal grammar $G$, in which case permissible representations will be restricted to strings of the formal language $L(G)$. For example, in the standard PCFG model the permissible analyses are defined by a context-free grammar $G$. But

[9] For a somewhat different but largely compatible view, see Collins (1999); cf. also Manning and Schütze (2000).

it can also be a model that provides constraints on representations without defining a string language in the process. A simple example would be a model that allows any context-free parse tree whose nonterminal nodes are labeled with symbols from a given set $N$ but whose terminal nodes are labeled with arbitrary word tokens occurring in text sentences of $L$. As pointed out by Bod (1998), such a model is not a grammar in the formal sense but a dynamic system, since the set of accepted strings cannot be defined independently of the input to the system.

The sample of text $T_t$, which will normally be called the *training data*, may or may not be annotated with representations satisfying the constraints of $M$, i.e., it may or may not be extracted from a *treebank* of the language $L$. If $T_t$ is a treebank sample, then there also exists a corresponding sequence of analyses $A_t = (y_1, \ldots, y_n)$, where $y_i$ is the correct analysis of $x_i$ according to the treebank annotation. Then the inductive inference scheme $I$ will typically be based on a form of *supervised machine learning* (Mitchell, 1997; Hastie et al., 2001). A typical example is the induction of a PCFG by extracting all context-free productions encountered in the treebank and using the relative frequency of each production to estimate its probability, a so-called *treebank grammar* (Charniak, 1996). If $T_t$ is a raw text sample, there is no sequence of analyses given, but *unsupervised learning* may be used. Thus, early work on PCFG parsing applied the Inside-Outside algorithm to raw text corpora in order to estimate the probabilistic parameters of a hand-crafted context-free grammar (Fusijaki et al., 1989; Pereira and Schabes, 1992). However, since the accuracy obtained with unsupervised methods remains inferior to supervised approaches, the latter have dominated the field in recent years.

The inductive inference scheme $I$, which defines the actual analyses of a given string $x$, relative to the model $M$ and the sample $T_t$, can often be decomposed into three distinct components:

1. A parameterized *stochastic model* $M_\Theta$ assigning a score $S(x, y)$ to each permissible analysis $y$ of a sentence $x$, relative to a set of parameters $\Theta$.
2. A *parsing method*, i.e., a method for computing the best analysis $y$ for a sentence $x$ according to $S(x, y)$ (given an instantiation of $\Theta$).
3. A *learning method*, i.e., a method for instantiating $\Theta$ based on inductive inference from the training sample $T_t$.

In the PCFG model, the parameters in $\Theta$ are the probabilities associated with the rules of the context-free grammar, while the score $S(x, y)$ is the joint probability $P(x, y)$, which can be computed by multiplying rule probabilities according to the normal independence assumptions. As we have already seen, this score can be used to rank alternative analyses and select the optimal analysis according to the model. As parsing method, we can use one of the standard context-free parsing algorithms extended to PCFG parsing, such as the CKY algorithm (Ney, 1991) or Earley's algorithm (Stolcke, 1995). As learning method, it is common to use some form of maximum likelihood estimation, either based on relative frequencies from a treebank, or based on the

unsupervised Inside-Outside algorithm. In any case, it is important to note that one and the same parameterized stochastic model $M_\Theta$ can be combined with different parsing methods as well as different learning methods, and that parsing methods and learning methods are largely independent of each other.

It is worth emphasizing that in order for the system to be usable in practice, there must be effective ways to implement parsing and learning methods, so that the actual analyses for a sentence can be computed with reasonable efficiency. Usually, this computation is divided into a *training phase*, where the learning method is applied once to the training sample $T_t$ in order to estimate the parameters of the model $M_\Theta$, and a *parsing phase*, where analyses are constructed and scored for individual sentences, although the exact division of labor between the phases depends on the methods involved. As noted in relation to the PCFG model, parsing methods in the data-driven approach are often closely related to grammar parsing algorithms, especially if the model $M$ is a formal grammar or some other model with a closely related structure. However, for certain types of models it may not be possible to use standard grammar parsing methods, because the search space defined by the stochastic model is too complex. We will return to this problem when we discuss the efficiency problem for the data-driven approach.

In the previous section, we observed that grammar-based text parsing rests on the assumption that the text language $L$ can be approximated by a formal language $L(G)$ defined by a grammar $G$. The data-driven approach is also based on an approximation, but this approximation is of an entirely different kind. While the grammar-based approximation in itself only defines permissible analyses for sentences and has to rely on other mechanisms for textual disambiguation, the data-driven approach tries to approximate the function of textual disambiguation directly. And while the grammar-based approximation is an essentially deductive approach, the data-driven approach is based on inductive inference from a finite sample $T_t = (x_1, \ldots, x_n)$ to the infinite language $L$.

It is a fundamental property of inductive inference that without making any a priori assumptions we have no rational basis for choosing one hypothesis over the other. For instance, there is an infinite number of languages $L$ that are compatible with any finite text sample $T_t$. Thus, in order to support any kind of generalization beyond the training sample $T_t$, the inference scheme $I$ must introduce an *inductive bias*, which can be defined as a minimal set of assertions $B$ such that our inferences are entailed by $B$ together with (a logical description of) the sample $T_t$ (Mitchell, 1997). The bias of a particular model will in general depend both on the stochastic model $M_\Theta$ and on the learning method used.

Choosing the right inductive bias is essential for a good approximation, and research on machine learning of natural language during the last ten to fifteen years has produced many results about what bias may be appropriate for different problems in natural language processing. We will review some of this research later in this chapter, when we discuss the problem of disambiguation

in data-driven parsing. For the time being, we will simply observe that whereas the grammar-driven approach depends on a more or less satisfactory language approximation, the data-driven approach depends on inductive inference from a more or less representative language sample using a more or less appropriate inductive bias. These different starting points explain why problems such as robustness, disambiguation, accuracy and efficiency may appear quite different in the two extreme approaches. Let us now proceed to an examination of these problems in the context of data-driven text parsing.

If the grammar-based approach is sometimes characterized as being strong with respect to accuracy, but weaker with respect to robustness, disambiguation and efficiency, the reverse is often said to be true for the data-driven approach. In both cases, this is at best an oversimplification. Starting with *robustness*, there is no reason that the data-driven approach should be inherently more robust than the grammar-based approach. It all depends on properties of the formal model $M$ as well as the inference scheme $I$ used for generalization to unseen sentences. However, it is a contingent fact about most existing data-driven systems for text parsing that these components are defined in such a way that any possible input string $x$ is assigned at least one analysis, which means that the robustness problem is eliminated.

This kind of absolute robustness can be illustrated by the framework of Data-Oriented Parsing (DOP) (Bod, 1995, 1998; Bod, Scha and Sima'an, 2003). More precisely, we will consider the model DOP3 in (Bod, 1998), which can be described as follows:

1. The formal model $M$ defines as a permissible analysis for a string $x$ any parse tree that can be composed from subtrees of trees in the text sample, using leftmost node substitution and allowing the insertion of words from $x$ (even if these do not occur in the training sample).
2. The text sample $T_t = (x_1, \ldots, x_n)$ is a sample of sentences from a treebank containing the corresponding context-free parse trees $A_t = (y_1, \ldots, y_n)$.
3. The inductive inference scheme $I$ is based on a stochastic model $M_\Theta$ that defines the probability $P(x, y)$ to be the sum of the probabilities $P(D_i)$, for every derivation $D_i$ of $y$ for $x$, which in turn is defined as the product of the probabilities $P(t_j)$, for every subtree $t_j$ used in $D_i$.

It is a fundamental property of this model that, since any subtree has a non-zero probability, and since parse trees can be composed from arbitrary subtrees, any input string $x$ can be assigned an analysis $y$ with a non-zero probability $P(x, y)$. This in turn means that, provided that the model can be paired with efficient learning and parsing methods, the robustness problem is eliminated.

A consequence of the extreme robustness is that these data-driven parsers will analyze strings that are probably not in the text language $L$ under any characterization. If we compare this to the grammar-driven language approximation, where the robustness problem arises from the fact that some sentences in $L$ are not in the language $L(G)$ defined by the grammar, we can say that

the data-driven approach avoids the robustness problem by a kind of superset approximation, i.e., any sentence in $L$ is a string that can be analyzed by the parser, but not necessarily vice versa.

Another consequence is that the number of analyses assigned to each input string will usually be very large. In this way, the development from PCFGs to robust DOP models can be seen as an extreme form of constraint relaxation, which is one of the methods used to alleviate the robustness problem in grammar-driven parsing but which can easily lead to an explosion in the number of analyses assigned to a sentence (cf. section 2.2.1). However, given a proper stochastic model, this is a rather controlled form of constraint relaxation, since all the different analyses can be ranked with respect to their score, which also means that we have a method for pruning the search space in a principled way, which is often necessary for reasons of efficiency.

Given the way in which the data-driven approach normally eliminates the robustness problem, there is little use for the second main technique to handle robustness in the grammar-driven approach, namely partial parsing. However, this does not mean that data-driven methods cannot be used to achieve partial parsing, e.g., chunking, as shown very early by Church (1988), using probabilistic methods, and later on by Ramshaw and Marcus (1995), using transformation-based learning. Other learning methods that have been used for data-driven partial parsing include memory-based learning (Veenstra, 1998; Tjong Kim Sang and Veenstra, 1999) and support vector machines (Kudo and Matsumoto, 2000). A collection of data-driven methods applied to partial parsing can be found in Cardie et al. (2000) and a survey of the field in Hammerton et al. (2002). Data-driven methods have also been used to construct cascaded partial parsers that approximate full parsing, e.g., using Hidden Markov Models (Brants, 1999) or memory-based partial parsing and grammatical relation finding (Argamon et al., 1998; Daelemans et al., 1999; Tjong Kim Sang and Veenstra, 2001; Buchholz, 2002).

As already noted, the problem of *disambiguation* can in many cases be even more severe in data-driven text parsing than for traditional grammar-driven systems, since the improved robustness that is the result of extreme constraint relaxation comes at the expense of massive overgeneration or leakage. However, this is compensated by the fact that the inductive inference scheme provides a mechanism for disambiguation, either by associating a score with each analysis, intended to reflect some optimality criterion, or by implicitly maximizing this criterion in a deterministic selection. In general, inductive learning methods can be grouped into three groups according to the type of mechanism they provide for ranking or selection (Jebara, 2004):

1. Generative models score analyses with the joint probability $P(x, y)$ of the string $x$ and the analysis $y$.
2. Conditional models score analyses with the conditional probability $P(y \mid x)$ of the analysis $y$ given the string $x$.

3. Discriminative models select the analysis $y$ that maximizes the conditional probability $P(y \mid x)$ (without computing it).

Conditional and discriminative models are not always distinguished in the literature, and the term *discriminative* is often used to cover all models that attempt to maximize the conditional probability $P(y \mid x)$. It is also important to distinguish the structure of the model and the probability that is maximized, which is what we are concerned with here, from the method of parameter estimation, which may also be classified as generative, conditional or discriminative (Klein and Manning, 2002; Henderson, 2004). We will return to this distinction later on.

In the previous section, we discussed some of the early work on data-driven disambiguation, based on the generative PCFG model. The relatively poor parsing accuracy achieved with this model is usually attributed to two major weaknesses: the lack of sensitivity to lexical dependencies and the lack of sensitivity to structural preferences (Collins, 1999).[10] The first observation has motivated a wide range of experiments with lexicalized probabilistic models (Hindle and Rooth, 1991; Schabes, 1992; Resnik, 1992; Collins and Brooks, 1995; Charniak, 1997a). And even though the significance of lexicalization has later been called into question by Klein and Manning (2003), it is striking that virtually all current models for data-driven disambiguation make use of lexical information. Many of these models, which are based on binary relations between lexical items, can be subsumed under the notion of *bilexical grammar* (Eisner, 2000). Bilexical relations are especially important in models based on dependency representations of syntactic structure, such as those of Eisner (1996a,b), Yamada and Matsumoto (2003), and the models investigated in this book (Nivre et al., 2004; Nivre and Scholz, 2004).

The second weakness of the PCFG model is the lack of sensitivity to structural preferences, such as the preference for right-branching or left-branching structures in different languages (Collins, 1999). In order to capture such preferences, we need a different parameterization of syntactic structures, with more flexible independence assumptions, so that the probability of a certain structure can be conditioned on the most significant parts of the surrounding structure. This was the main motivation for the development of *history-based* models of natural language processing, which were first introduced by Black et al. (1992) and have been used extensively both for tagging and parsing. The idea is to map each pair $(x, y)$ of an input string $x$ and an analysis $y$ to a sequence of decisions $D = (d_1, \ldots, d_n)$. In a generative model, the joint probability $P(x, y)$ can then be expressed using the chain rule of probabilities as follows:

$$P(x, y) = P(d_1, \ldots, d_n) = \prod_{i=1}^{n} P(d_i \mid d_1, \ldots, d_{i-1}) \tag{2.1}$$

[10] Further problems with the PCFG model are discussed by Briscoe and Carroll (1993); cf. also Klein and Manning (2003).

The conditioning context for each $d_i$, $(d_1, \ldots, d_{i-1})$, is referred to as the *history* and usually corresponds to some partially built structure. In order to get a tractable learning problem, histories are then grouped into equivalence classes by a function $\Phi$ (Black et al., 1992):

$$P(x, y) = P(d_1, \ldots, d_n) = \prod_{i=1}^{n} P(d_i \mid \Phi(d_1, \ldots, d_{i-1})) \tag{2.2}$$

Early versions of this scheme were integrated into grammar-driven systems. For example, Black et al. (1993) used a standard PCFG but could improve parsing performance considerably by using a history-based model for bottom-up construction of leftmost derivations. Briscoe and Carroll (1993) instead started from a unification-based grammar and employed LR parsing, using supervised learning to assign probabilities to transitions in an LALR(1) parse table constructed from the context-free backbone of the original grammar (cf. also Carroll and Briscoe, 1996).

Generative history-based models are most well-known from the influential work of Collins (1997, 1999) and Charniak (2000). These models are based on a stochastic process generating parse trees top-down, where the children of a given node are generated, not by complete productions as in the PCFG model, but by Markov processes generating one child at a time, conditioned on some partially built structure. Charniak (1997b) uses the popular term *Markov grammar* for this kind of model, although these models are not strictly speaking grammar-driven, since they will normally accept any input string in the same way as the DOP model considered earlier. The Collins and Charniak models are also similar in that they make heavy use of lexical dependencies, a property inherited from their earlier models (Collins, 1996; Charniak, 1997a). Generative models of a similar kind have also been proposed for dependency-based syntactic representations, e.g., by Eisner (1996a,b) and Wang and Harper (2004).

The various models proposed within the DOP framework (Bod, 1995, 1998; Bod, Scha and Sima'an, 2003) are usually not considered to be history-based, although they are normally generative models. However, it is not hard to relate them to the history-based approach. The first difference is that the DOP model assumes a many-to-one relationship between derivations and analyses, which means that the probability $P(x, y)$ has to be computed as a sum over all derivations $D_i = (d_1, \ldots, d_{n_i})$ of the analysis $y$ for the input string $x$ (assuming that $D_1, \ldots, D_m$ are all the derivations of analysis $y$ for input $x$):

$$P(x, y) = \sum_{i=1}^{m} P(d_1, \ldots, d_{n_i}) = \sum_{i=1}^{m} \prod_{j=1}^{n_i} P(d_j \mid \Phi(d_1, \ldots, d_{j-1})) \tag{2.3}$$

The second difference is that derivations are defined only in terms of which fragments are used to build the analysis, where fragments are assumed to be independent of each other:

$$P(x,y) = \sum_{i=1}^{m} P(d_1, \ldots, d_{n_i}) = \sum_{i=1}^{m} \prod_{j=1}^{n_i} P(d_j) \tag{2.4}$$

The idea of summing over all derivations appears to be good for disambiguation but makes parsing intractable (Bod, 1998, chapter 4), a problem to which we will return later in this section. The DOP models have a lot in common with the stochastic version of LTAG parsing, which is also based on a sum-of-products model, albeit in combination with a grammar-based approach where permissible fragments and composition operations are defined by the LTAG formalism (Joshi and Sarkar, 2003).

History-based models can also be used to define conditional models, where the pair $(x, y)$ is still modeled as a sequence of decisions but where the input string $x$ is a conditioning variable:

$$P(y \,|\, x) = P(d_1, \ldots, d_n \,|\, x) = \prod_{i=1}^{n} P(d_i \,|\, \Phi(d_1, \ldots, d_{i-1}, x)) \tag{2.5}$$

Conditional history-based models were used by Jelinek et al. (1994), in the first system that did not make use of a hand-crafted grammar but extracted all the necessary information from treebank data, and by Magerman (1995), in the first system that showed a significant improvement over systems based on the standard PCFG model. While both these systems used probabilistic decision trees for parameter estimation, later versions of the conditional history-based approach are mostly based on maximum entropy models (Berger et al., 1996; Della Pietra et al., 1997), also known as exponential or log-linear models, as pioneered by Ratnaparkhi (1997, 1999). Conditional history-based models for dependency-based representations have been investigated in Eisner (1996a,b).

Whereas the early conditional models, including that of Ratnaparkhi (1997, 1999), only scored individual parsing decisions in isolation, later work has extended the idea to models that score complete analyses. These models are often used to rerank a small set of analyses produced by another model, typically a generative model, as in the work of Johnson et al. (1999); Collins (2000); Collins and Duffy (2002); Collins and Koo (2005); Charniak and Johnson (2005). This type of model has also been used very successfully for parse selection in grammar-driven frameworks based on linguistic theories such as LFG (Riezler et al., 2002), HPSG (Toutanova et al., 2002; Miyao et al., 2003), and CCG (Clark and Curran, 2004).

Conditional models, such as the maximum entropy model, are normally combined with conditional parameter estimation, i.e., estimation procedures that try to maximize the conditional likelihood of the training data. But conditional estimation can also be used with generative models. Thus, Johnson (2001) obtained a small (non-significant) improvement for a standard PCFG model, using maximum conditional likelihood estimation instead of the traditional maximum joint likelihood estimation. More recently, Henderson (2004) has shown that a history-based generative model based on left-corner parsing,

combined with conditional parameter estimation, outperforms both the generative model with joint estimation and a conditional model with conditional estimation. These results are in line with the argument of Klein and Manning (2002) to the effect that whereas conditional estimation methods often have an advantage over joint estimation, conditional model structure may in fact be harmful in many cases (although the empirical results in their article concern part-of-speech tagging rather than parsing).

There are also purely discriminative versions of the history-based model, i.e., models that implicitly try to maximize the conditional probability of each parsing decision without actually computing it. These models are usually combined with deterministic processing, since discriminative learning does not support a probabilistic scoring and ranking of complete analyses. In this category, we find most data-driven methods for partial parsing based on discriminative learning and deterministic left-to-right processing discussed earlier (Argamon et al., 1998; Daelemans et al., 1999; Kudo and Matsumoto, 2000; Tjong Kim Sang and Veenstra, 2001; Buchholz, 2002). Although these methods are seldom associated with the notion of history-based parsing in the literature, they do in fact implement a history-based strategy, classifying segments from left-to-right while basing their decisions on a combination of input features and previously classified segments.

Discriminative history-based methods for full parsing have been proposed for dependency-based representations by Yamada and Matsumoto (2003), who use a form of shift-reduce parsing for dependency-based representations in combination with support vector machines (Vapnik, 1995). Within the framework investigated in this book, Nivre et al. (2004) and Nivre and Scholz (2004) use a similar parsing algorithm but rely on memory-based learning (Daelemans and Van den Bosch, 2005) to predict parser actions. These frameworks are similar to early conditional parsing models, such as Ratnaparkhi (1997, 1999), in that inductive learning applies to individual parsing actions in isolation. A more holistic discriminative approach to full parsing can be found in the memory-based framework of Kübler (2004), where new analyses are constructed from arbitrarily large fragments of analyses for similar sentences in the training data, in a way which has close affinities with the DOP framework. Another close relative of DOP based on memory-based learning can be found in De Pauw (2003).

The problem of *accuracy* has been implicit throughout the discussion of disambiguation in this section. Since the data-driven approach, as defined here, always provides a mechanism for disambiguation, whether stochastic or deterministic, it is usually trivial to fulfill the requirement of disambiguation as such. Hence, research in this area during the last ten to fifteen years has mainly been focused on improving the accuracy of disambiguation. However, it is also worth noting that, because the data-driven approach incorporates an optimization criterion in the training phase, whether explicitly or implicitly, it is possible to optimize models for different criteria of accuracy. The relationship between estimation objectives and evaluation metrics has been the

subject of several studies (Goodman, 1996, 1998; Johnson, 2001; Klein and Manning, 2002; Sima'an, 2003).

With respect to the final problem of *efficiency*, the conventional wisdom seems to be that the data-driven approach is superior to the grammar-driven approach, but often at the expense of less adequate output representations (Kaplan et al., 2004). However, in reality we find as much variation among data-driven approaches as among grammar-driven approaches, and the overall picture is in fact very similar.

At one end of the scale, we find frameworks where the parsing problem is computationally intractable, such as the original DOP model (Sima'an, 1996a, 1999). Research on efficient parsing within the DOP framework has therefore focused on finding efficient approximations that preserve the advantage gained in disambiguation by considering several derivations of the same analysis. While early work focused on a kind of randomized search strategy called Monte Carlo disambiguation (Bod, 1995, 1998), the dominant strategy has now become the use of different kinds of PCFG reductions (Sima'an, 1996b; Goodman, 1996; Bod, 2001, 2003).

At the other end of the scale, we find highly efficient methods that perform parsing in linear time, such as the various deterministic methods for partial parsing. However, linear-time processing is achievable also for full parsing, either as a theoretical worst case (Nivre, 2003) or as an empirical average case (Ratnaparkhi, 1997, 1999).

In between, we find parsers based on history-based probabilistic models, whether generative or conditional, where parsing in principle consists in deriving all the possible analyses for a given input string and selecting the optimal analysis with respect to the probabilistic model. In this case, there is often a trade-off between accuracy in disambiguation and efficiency in processing. Broadly speaking, the more sophisticated models proposed in recent years have generally led to more accurate disambiguation but less efficient processing.

For example, with the standard PCFG model all possible analyses can be constructed in $O(n^3)$ time using a standard algorithm for context-free parsing. Moreover, given the independence assumptions of this model, the selection of the most probable analysis can be integrated into the parsing process using Viterbi optimization (Viterbi, 1967), as shown for the CKY algorithm by Ney (1991) and for Earley's algorithm by Stolcke (1995).

With the more complex history-based models, such as Collins (1997, 1999) and Charniak (2000), parsing becomes less efficient for two reasons. First, the drastic constraint relaxation leads to an explosion in the number of possible analyses for any given input string. Secondly, the more complex probability model does not allow the same reduction of the search space as the standard Viterbi algorithm for PCFGs. This means that, even if parsing does not become intractable, the time complexity may be such that an exhaustive search of the analysis space is no longer practical. For example, the complexity of the parsing algorithm used in Collins (1999) is $O(n^5)$. In practice, most systems

of this kind only apply the full probabilistic model to a subset of all possible analyses, resulting from a first pass based on an efficient approximation of the full model. This first pass is normally implemented as some kind of chart parsing with beam search, using an estimate of the final probability to prune the search space (Caraballo and Charniak, 1998). Another strategy for reducing the set of candidate analyses is to rely on an initial supertagging phase, as in the grammar-driven CCG approach (Clark and Curran, 2004; Curran and Clark, 2004).

Finally, for data-driven approaches the time required for training, although less critical than parsing time, should also be taken into consideration when discussing efficiency. For instance, learning methods that rely on numerical optimization, such as maximum entropy modeling, may require repeatedly reparsing the training corpus with the current model to determine the parameter updates that will improve the training criterion. In this respect, lazy learning methods such as memory-based learning have a clear advantage, since they reduce learning to the efficient storage of training instances. However, this of course means that more processing has to take place at parsing time, which may limit the advantage.

### 2.3.3 Converging Approaches

We have now considered two different strategies for text parsing, the grammar-driven approach and the data-driven approach. Although these strategies have different points of departure, we have seen that they in practice often lead to similar solutions when confronted with the mutually interacting requirements of robustness, disambiguation, accuracy and efficiency. In fact, our characterization of the two approaches is such that many contemporary frameworks for text parsing instantiate both. As illustrated in figure 2.6, we can distinguish approaches that are grammar-driven but not data-driven, such as CG (Karlsson, 1990; Karlsson et al., 1995) and its descendant FDG (Tapanainen and Järvinen, 1997; Järvinen and Tapanainen, 1998), and approaches that are data-driven but not grammar-driven, such as different varieties of history-based parsing (Ratnaparkhi, 1997, 1999; Collins, 1997, 1999; Charniak, 2000), as well as most incarnations of Data-Oriented Parsing (Bod, 1995, 1998; Bod, Scha and Sima'an, 2003). But we also find many frameworks that combine the use of formal grammars with data-driven methods to achieve robustness and disambiguation, such as broad-coverage parsers based on PCFG (Black et al., 1993), LTAG (Bangalore and Joshi, 1999), LFG (Riezler et al., 2002), HPSG (Toutanova et al., 2002; Miyao et al., 2003), and CCG (Clark and Curran, 2004).

Before we close the discussion of grammar-driven and data-driven text parsing, it may be useful to relate these concepts to a few other conceptual distinctions that are often made in the literature on natural language parsing. The first is the distinction between *deep parsing* and *shallow parsing*, which

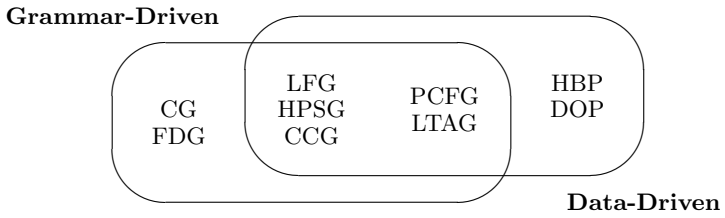


**Fig. 2.6.** Convergence of grammar-driven and data-driven text parsing

has to do with the amount of information contained in the syntactic representations produced by the parser, where deep parsing typically refers to the kind of representations found in linguistic theories like LFG (Kaplan and Bresnan, 1982), HPSG (Pollard and Sag, 1994) or CCG (Steedman, 2000), while shallow parsing can be exemplified with the skeletal constituent structures found in the first version of the Penn Treebank annotation scheme (Marcus et al., 1993) or the partial specification of grammatical functions in CG (Karlsson, 1990; Karlsson et al., 1995). In other words, this is a distinction between different kinds of syntactic representations and should not be confused with a distinction between different parsing methods. While it is true that most systems for deep parsing are grammar-based, there also exist data-driven approaches to deep parsing, such as the LFG-DOP model (Bod and Kaplan, 1998). And shallow parsing can be performed with grammar-driven as well as data-driven methods.

The second distinction is that between *full parsing* and *partial parsing*, which we have already touched upon several times. This distinction, which is sometimes confused with the previous distinction between deep and shallow parsing, has to do with the completeness of the analysis and can therefore only be defined relative to a specific target representation. Thus, a segmentation of the input string into base chunks can be regarded as a partial specification of a constituent analysis, and an assignment of grammatical functions to individual word tokens can be seen as a partial specification of a dependency structure. This means that both deep and shallow parsing can be implemented as full or partial parsing and that either grammar-driven or data-driven methods may be used in the realization.[11]

The third and final distinction is the distinction between *rule-based* and *example-based* (or *analogical*) language processing, which has to do with whether the analysis of new sentences is based on abstract rules (in a wide

[11] There is a further complication in that the term *partial parsing* is used both about fallback strategies adopted to cope with the robustness problem in full parsing (cf. section 2.3.1) and about approaches where the final goal is always a partial analysis, such as *chunking*. It is only partial parsing in the latter sense that is sometimes referred to as *shallow parsing*.

sense) or on concrete representations (as found in an annotated corpus). Whereas grammar-driven approaches are by necessity rule-based, data-driven approaches can involve more or less abstraction. Thus, most probabilistic parsing models are similar to grammars in that they are abstract generalizations over sets of concrete representations. This is true not only of models that involve a formal grammar, like the PCFG model, but also of most history-based models for parsing, which use training data to estimate the parameters of an abstract probabilistic model.[12] This is in contrast to the DOP framework, where concrete examples in the training corpus are stored as is and used as the raw material for constructing new analyses during processes, thus implementing an example-based approach to syntactic parsing. The same is true of memory-based approaches to parsing, such as Kübler (2004).

One of the conclusions that we want to draw from the discussion in this chapter is that the partly conflicting requirements of robustness, disambiguation, accuracy and efficiency give rise to a complex optimization problem, which we can try to solve in different ways but which always requires a joint optimization. The wide variety of different methods for text parsing can to a large extent be said to result from different optimization strategies and different goals.

The grammar-driven approach, in its purest form, starts from a system with optimal accuracy, in the sense that only sentences for which the correct analysis can be derived are covered, and gradually seeks to improve the system with respect to robustness and disambiguation. However, this development may compromise efficiency, which therefore has to be optimized together with robustness and disambiguation.

By contrast, the data-driven approach, in its most radical form, starts from a system with optimal robustness and disambiguation, in the sense that every sentence gets exactly one analysis, and gradually seeks to improve the system with respect to accuracy. Again, this may lead to problems of efficiency, which therefore has to be optimized together with accuracy. In both cases, we may furthermore put more or less priority on efficiency in relation to other optimization requirements.

When evaluating different strategies, we are of course interested in whether they lead to optimal performance overall. However, because of the interaction between different requirements, we can only define the optimum with respect to a given prioritization. It is true that accuracy is of prime importance, but if optimal accuracy leads to computational intractability, its practical interest is limited. Moreover, from a scientific point of view, we need to increase our knowledge about the complex interaction of conflicting requirements, which means that studies based on different goals and strategies can be valuable,

[12] The term *Markov grammar* for this type of model (Charniak, 1997b) is motivated by this similarity, even though the model normally does not involve a grammar in the formal sense (cf. section 2.3.2).

even if they do not always advance the state of the art in terms of raw performance (on any of the dimensions considered here).

### 2.3.4 Inductive Dependency Parsing

It seems appropriate to conclude this discussion of methods for text parsing by situating the framework of inductive dependency parsing in the wider landscape of different parsing methods, grammar-driven and data-driven, deep and shallow, full and partial, rule-based and example-based.

Inductive dependency parsing is an instance of the data-driven approach to text parsing, which in this study is combined with a rather extreme optimization strategy. The first step in this strategy is to construct a framework that is provably optimal with respect to robustness, disambiguation and efficiency. The second step is to use inductive learning to gradually improve parsing accuracy without sacrificing robustness, disambiguation or efficiency. The first step will be taken in chapter 3, while the second step will occupy us throughout chapters 4–5.

Robustness will be achieved with the kind of radical constraint relaxation found in many other data-driven approaches, which means that some analysis will be assigned to every input string. However, we will not resort to partial parsing but instead construct complete dependency representations for every input string, even though these representations will normally be rather shallow syntactic representations. Our task could therefore be characterized as full parsing with shallow dependency representations.

The use of dependency representations, which is a central element of the framework, will be discussed in more detail in chapter 3. It will be argued that, over and above any other motivation for using these representations, dependency structures provide the right level of underspecification to allow robust and efficient full parsing.

Disambiguation will be performed by deterministic processing in combination with discriminative learning, using a history-based model for parsing decisions. Although the general approach is compatible with many different learning methods, we will concentrate in this book on the use of memory-based learning of parsing decisions, which means that the approach can also be described as example-based, or analogical, rather than rule-based.

Efficiency in parsing is gained mainly through the use of deterministic processing, which means that parsing can be performed in linear time, which is arguably optimal since any reasonable parsing method will at least have to scan the input from start to end. Moreover, the use of a lazy learning method also gives good efficiency in training.

Given our overall optimization strategy, it is important to have precise evaluation criteria for the basic requirements of robustness, disambiguation, accuracy and efficiency. The last section of this chapter will be devoted to the definition of such criteria.

## 2.4 Evaluation Criteria

Most of the discussion in this chapter applies to natural language parsing in general and not only to the particular framework investigated in this book. However, when we come to concrete evaluation criteria, it is important that these reflect the general strategy underlying our research efforts. We will therefore concentrate in this section on evaluation criteria that are relevant for the version of inductive dependency parsing developed in this book, criteria that may in some respects be less suitable for other approaches to text parsing.

### 2.4.1 Robustness

Following the discussion in sections 2.3.1–2.3.2, we regard robustness in text parsing as the capacity of a system to analyze any input sentence. In other words, we do not attempt to draw any distinction between (lack of) coverage and (lack of) robustness. In this respect, we agree with Chanod (2001) that robustness is about exploring all constructions humans actually produce in natural language texts, 'be they grammatical, conformant to formal models, frequent or not' (Chanod, 2001, 188). This is admittedly a rather weak notion of robustness, which is easy to achieve in itself, but which can be considered a prerequisite for the optimization of other criteria. Stronger notions of robustness discussed in the literature, which relate to the degradation of accuracy with increasingly ill-formed input (Menzel, 1995; Basili and Zanzotto, 2002), can be regarded as additional requirements on top of the basic requirement of producing some analysis for every input.

Since we want the requirement of robustness to be independent of other requirements, in particular the requirement of disambiguation, we propose the following definition:

**Definition 2.1.** A system $P$ for parsing texts in language $L$ satisfies the requirement of *robustness* if and only if, for any text $T = (x_1, \ldots, x_n)$ in $L$, $P$ assigns *at least* one analysis to every text sentence $x_i \in T$.

Given our optimization strategy, we want to treat robustness as an absolute requirement, which means that we will require formal proofs of this property. In other frameworks, where robustness is an optimization criterion rather than an absolute requirement, it may be more relevant to perform an empirical evaluation of the degree to which robustness can be achieved for representative input texts.

### 2.4.2 Disambiguation

Disambiguation in text parsing implies the selection of a single analysis for every text sentence from all the analyses that are compatible with the input string according to the formal framework adopted. From this perspective, it is irrelevant whether the input string is genuinely ambiguous or whether the

fact that there are several candidate analyses is due to the so-called leakage problem. It is also irrelevant whether the selection is achieved through an $n$-best ranking of a large space of candidate analyses or through a deterministic procedure that only constructs a single analysis for a given input sentence. More precisely, we define the requirement of *disambiguation* in the following way:

**Definition 2.2.** A system $P$ for parsing texts in language $L$ satisfies the requirement of *disambiguation* if and only if, for any text $T = (x_1, \ldots, x_n)$ in $L$, $P$ assigns *at most* one analysis to every text sentence $x_i \in T$.

Since we want disambiguation to be an absolute requirement for our parsing methods, in the same way as robustness, we will require formal proofs of this property as well. In another context, it would be possible to regard disambiguation as a desideratum rather than a hard requirement, and to perform an empirical evaluation of a system's capacity for disambiguation when applied to representative samples of text.

The requirements of robustness and disambiguation are independent of each other, but any system satisfying both requirements must assign exactly one analysis to every text sentence. This is the notion of *robust disambiguation* that we will adopt as an absolute requirement for inductive dependency parsing.

### 2.4.3 Accuracy

Requiring a single analysis for every sentence in a text is obviously of limited interest unless we also require the analysis to be correct. According to the characterization of text parsing in section 2.2.2, we assume that there exists a single correct analysis for each sentence in a text. An absolute requirement of *accuracy* could therefore be stated as follows:[13]

**Definition 2.3.** A system $P$ for parsing texts in language $L$ satisfies the requirement of *accuracy* if and only if, for any text $T = (x_1, \ldots, x_n)$ in $L$, $P$ assigns the correct analysis to every text sentence $x_i \in T$.

However, given the state of the art in natural language parsing, complete accuracy has to be regarded as an asymptotic goal, impossible to attain in practice for any non-trivial domain of natural language text, and especially in combination with robustness and disambiguation. Moreover, even if perfectly accurate parsing were possible, there would be no formal method for proving that a system satisfies this requirement, since the language $L$ is not a formal language. This means that we must always rely on empirical evaluations using representative samples of text from the given language $L$.

[13] In a context where it is not assumed that every sentence has exactly one correct analysis, an alternative would be to replace 'the correct analysis' with 'at least one correct analysis'. In this study, the two formulations are practically equivalent, given the absolute requirement of robust disambiguation.

Given the optimization strategy adopted in this study, accuracy should be treated as an optimization criterion, where the goal is to maximize accuracy while maintaining robustness and disambiguation. We will adopt the standard methodology for evaluating accuracy in text parsing, which is to treat samples of treebank data for the language $L$ as an empirical gold standard and use inferential statistics to generalize the results to arbitrary texts in $L$. As noted in section 2.2.2, this methodology is problematic in several ways, but we will postpone a discussion of these problems until chapter 5. There we will also define the different evaluation metrics that will be used for the evaluation of accuracy.

### 2.4.4 Efficiency

Finally, in order for methods to be usable in practical applications, they must allow tractable computation, which leads to the requirement of *efficiency*. As is customary, we will restrict ourselves to the computational resources of *time* and (to a lesser extent) *space*. In theory, tractability is usually taken to mean computable in deterministic polynomial time and space (relative to the size of the input), but many practical applications put higher demands on efficiency. Our strategy for optimization implies that we employ parsing algorithms with optimal complexity, which means that both time and space complexity should be linear in the size of the input. Thus:

**Definition 2.4.** A system $P$ for parsing texts in language $L$ satisfies the requirement of *efficiency* if and only if, for any text $T = (x_1, \ldots, x_n)$ in $L$, $P$ processes every text sentence $x_i \in T$ in time and space that is linear in the length of $x_i$.

Although the theoretical complexity of an algorithm can be proven formally, practical efficiency also depends on other factors and is therefore another optimization criterion that can be maximized while maintaining robustness and disambiguation. As discussed in section 2.3.2, there is often a tradeoff between accuracy and efficiency in text parsing, which means that we need to explore the joint optimization of accuracy and efficiency.

Given our strategy for optimization, we need to evaluate efficiency in two different ways. On the one hand, we will provide formal proofs of time and space complexity, establishing asymptotic bounds on worst-case running time and memory requirements. On the other hand, we will perform empirical experiments using treebank data, measuring actual running time and memory consumption.