Two-level probabilistic grammars for natural language parsing
Infante Lopez, G.G.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 1

Introduction

Natural language is a very complex phenomenon. Undoubtedly, the sentences we utter are organized according to a set of rules or constraints. In order to communicate with others, we have to stick to these rules up to a certain degree. This set of rules, which is language dependent, is well-known to all speakers of a given language, and it is this common knowledge that makes communication possible. Every sentence has a clear organization: words in an utterance glue together to describe complex objects and actions. This hidden structure, called syntactic structure, is to be recovered by a parser. A parser is a program that takes a sentence as input and tries to find its syntactic organization. A parser searches for the right structure among a set of possible analyses, which are defined by a grammar. The language model decides what the syntactic components of the sentence are and how they are related to each other, depending on the required level of detail.

Natural language parsers are used as part of many applications that deal with natural language. Applications like question answering, semantic analysis, speech recognition, etc. may rely heavily on parsers. The degree of detail in the information output by the parser may change according to the application, but some amount of parsing plays a role in many language technology applications, and parser performance may be crucial for the overall performance of the end-to-end application.

Designing and building language models is not a trivial task; the design cycle usually comprises designing a model of syntax, understanding its underlying mathematical theory, defining its probability distribution, and finally, implementing the parsing algorithm. The building of every new language model has to complete at least these steps. Each is very complex and constitutes a line of research in itself. To help handling the intrinsic complexity of these steps a sufficient level of abstraction is required. Abstraction is important as it helps us deal with complex objects by representing them with a subset of their characteristic features. The selected features characterize the object
for a given task or context. For example, the way we understand or see cars depends very much on the task we want to carry out with them: if we want to drive them, we do not need to know how their engines work, whereas if we are trying to fix a mechanical problem we better do. Moreover, abstraction has proven to be an important scientific principle. The way abstraction helps humans in dealing with complex systems can best be illustrated by the history of computer science. The complexity of systems has increased hand in hand with the introduction of programming languages that allow for increasing levels of abstraction, climbing all the way from machine code to assembly language to imperative languages such as Pascal and C, to today’s object-oriented languages such as Java.

Back to natural language parsing — what does abstraction have to do with parsing? Our view is that state-of-the-art natural language models lack abstraction: their design is often ad hoc, and they mix many features that, at least conceptually, should be kept separated. In this thesis, we explore new levels of abstraction for natural language models. We survey state-of-the-art probabilistic language models looking for characteristic features, and we abstract away from these features to produce a general language model. We formalize this abstract language model, establish important properties of the models surveyed, and with our abstract model we investigate new directions based on different parameterizations.

### 1.1 Probabilistic Language Models

Roughly speaking, the syntactic analysis of natural language utterances aims at the extraction of linguistic structures that make explicit how words interrelate within utterances. Syntactic structures for a sentence \( x \) are usually conceived as trees \( t_1(x), \ldots, t_n(x) \) whose leaves form the sentence \( x \) under consideration. A grammar is a device that specifies a set of trees. The trees in this set are said to be grammatical trees. Indirectly, the concept of a grammatical sentence is defined as follows: a sentence \( x \) is said to be grammatical if there is a grammatical tree that yields \( x \).

Most natural language grammars tend to assign many possible syntactic structures to the same input utterance. In such situations, we say that the sentence is ambiguous. Ambiguity is the most important unsolved problem that natural language parsers face. This contrasts with human language processing, which in most cases selects a single analysis as the preferred one for a given utterance. The task of selecting the single analysis that humans tend to perceive for an input utterance — disambiguation — is an active area of research in the field of natural language processing. Because of the roles of world knowledge, cultural preferences and other extra-linguistic factors, disambiguation can be seen as a decision problem under uncertainty. In recent
years, there have been different proposals for a solution, mainly based on probabilistic models. Probabilistic models assign probabilities to trees and then disambiguate by selecting the tree with the highest probability.

From a formal language perspective, the notion of a language coincides with the formal notion of a set of strings. Probabilistic languages extend this definition so that a language is a probability distribution over a set of trees. In particular, a probabilistic language model is a probability distribution over a set of utterance-analysis pairs. Usually, a recursive generative grammar is used to describe a set of possible utterance-analysis pairs, possibly allowing multiple pairs for the same utterance. Crucially, defining a probabilistic language model allows us to view disambiguation as an optimization task, where the most probable analysis $T^*$ is selected from among those that a grammar $G$ generates together with an input utterance $U$. If $P$ is a probability function over utterance-analysis pairs, i.e., a language model, we may describe this optimization task as follows:

$$T^* = \arg\max_{T \in G} P(T|U)$$

$$= \arg\max_{T \in G} \frac{P(T,U)}{P(U)}$$

where $\arg\max_{x \in X} f(x)$ stands for the $x \in X$ such that $f(x)$ is maximal, and where $P(U)$ is the same for all trees and, consequently, can be left out.

The step of enriching a given generative grammar with probabilities is a non-trivial task. Apart from the empirical question of how to do so in a way that allows good disambiguation, in the sense that the model selects the same preferred analysis as humans do, there are various formal and practical issues concerning the definition of a correct model. In order to fully understand a language model, it is necessary to abstract away from specific peculiarities and to identify its relevant features. The latter can be summarized as follows.

**Set of possible trees**: For a given utterance, the language model chooses a tree that articulates the syntactic structure of the utterance from a fixed set of possible trees. For example, in a Context Free Grammar (CFG), this set is defined by the grammar's tree language.

**Probabilities** Probabilistic language models assign a probability value to each tree, and this probability value is then used as a way to filter out unwanted trees. A significant part of the definition of a language model is used to establish the way probabilities are assigned to trees. For example, in a probabilistic context free
grammar, each rule has a probability value associated to it, and the probability of a tree is defined as the product of the probabilities assigned to the rules building up the tree.

**Parameter estimation** Probabilistic models contain many parameters that define the grammar's disambiguation behavior. These parameters have to be estimated. The probability model specifies the set of values the parameters might take. This, in turn, defines the set of possible grammars.

**Expressive power** The expressive power of language models gives us an idea of their algorithmic complexity, and it allows us to compare different models. For probabilistic models, determining their expressive power is a difficult job because the parameter estimation algorithm has to be taken into account.

**Tweaking** State-of-the-art parsing algorithms are not just language models — they have been optimized considerably in order to improve their performance on real natural language sentences. Some of the optimized parameters are hard to model and are usually outside the language model.

**Parsing complexity** Parsing complexity has become an issue again in recent years, because of the appearance of theoretically appealing models that seem very hard to implement efficiently. Parsing complexity should be as low as possible. The aim is to emulate the apparently linear time humans take to process a sentence.

Addressing all of these items in a single thesis would be overly ambitious. This list is meant to provide the context — throughout this thesis we study various specific aspects against this general background.

### 1.2 Designing Language Models

Designing a parsing algorithm involves a sequence of decisions about:

1. the grammatical formalism,
2. a probabilistic version of the formalism,
3. techniques for estimating probabilities, and
4. a parsing algorithm.

This cycle can be seen almost everywhere in the parsing literature (Collins, 1999; Eisner, 2000; Bod, 1998; Charniak, 1999; Ratnaparkhi, 1999). It seems that every interesting new parser uses a new formalism. The design is time consuming, and usually
parsers are only evaluated empirically. It is clear, however, that better empirical results do not necessarily convey a better understanding of the parsing problem. Usually, descriptions of state-of-the-art language models do not clearly state how the features presented in the previous section are defined or implemented, or how they fit in the design cycle. The grammatical framework is rarely the only decision responsible for the parsers's performance; on top of the decision, there is a lot of tweaking involved. For example, Collins (1997) defines a simple formalism, but in order to achieve his results, all the tweaking reported in (Bikel, 2004) is needed.

Designers of parsers often conflate their decisions about the different features we identify in Section 1.1. For example, let us zoom in on one of the characteristics identified in Section 1.1 and then step back to adopt a more abstract perspective, and see what this gives us. In the definition of today's state-of-the-art language models, Markov chains, and more specifically n-grams, are widely used because they are easy to specify and their probabilities easy to estimate. N-grams are both a component in the definition of a model and a technique to assign probabilities. They are central to language models, and, consequently, every property of language models must be evaluated with respect to n-grams. It might be helpful, though, to step back from n-grams, and think of them as special cases of probabilistic regular languages. Mathematical properties of probabilistic regular languages are as well understood as those of n-grams, but they fit more directly into the overarching theory of formal languages. This perspective allows us to clearly separate the definition of the model (using regular languages) and the procedure for estimating probabilities (using probabilistic regular language induction techniques).

In this thesis, we propose a language modeling formalism that abstracts away from any particular instance. We investigate three state-of-the-art language models and discover that they share a very noticeable feature: the set of rules they use for building trees is built on the fly, meaning that the set of rules is not defined a priori. The formalisms we review have two different levels of derivations even though this is not explicitly stated. One level is for generating the set of rules to be used in the second step, and the second step is for building the set of trees that characterize a given sentence. Our formalism, based on Van Wijngaarden grammars (W-grammars), makes these two levels explicit.

1.2.1 W-Grammars as the Backbone Formalism

W-grammars were introduced in the 1960s by Van Wijngaarden (1965). They are a very well-known and well understood formalism that is used for modeling programming languages (Van Wijngaarden, 1969) as well as natural languages (Perrault, 1984). W-grammars have been shown to be equivalent to Turing machines (Perrault, 1984),
Chapter I. Introduction

which are more powerful than we need: most state-of-the-art language models use grammatical formalisms that are much closer to context freeness than to Turing machines. In this thesis, we constrain the set of possible W-grammars in order to come closer to the expressive power of these grammatical formalisms. We denote this constrained version as CW-grammars.

Originally, W-grammars did not use probabilities, but part of the work presented below extends the formalism with probabilities. In this way, we define probabilistic CW-grammars (PCW-grammars). We show that probabilities are an essential component of the resulting formalism, not only because of the statistical perspective they bring, but also because of the expressivity they add. With PCW-grammars, we prove that Markovian context free grammars (Collins, 1999; Charniak, 1997), bilexical grammars (Eisner, 2000) and stochastic tree substitution grammars (Bod, 1998) are particular instances of probabilistic CW-grammars. The probabilistic version of CW-grammars helps us to prove properties for these models that were previously unknown.

1.3 Formal Advantages of a Backbone Formalism

From a theoretical point of view, general models help us to clarify the set of parameters a particular instance has fixed, and to make explicit assumptions that underlie a particular instance. It might be the case that these assumptions are not clear, or that, without taking the abstract model into account, the designer of a particular instance is completely unaware of them.

The role of probabilities: Our approach to parsing comes from a formal language perspective: we identify features that are used by state-of-the-art language models and take a formalism off the shelf and modify it to incorporate the necessary features. When analyzing the necessary features from the formal language perspective, the need for probabilities and their role in parsing are the first issue to address. In Chapter 3, we answer many questions regarding the role of probabilities in probabilistic context free grammars. We focus on these grammars because they are central to the formalism we present.

Consistency properties: General models do not add anything per se. Their importance is rather in the set of instances they can capture and the new directions they are able to suggest. In Chapter 4, we show that bilexical grammars, Markovian context free grammars and stochastic tree substitution grammars are instances of our general model. Our model has well-established consistency properties which we use to derive consistency properties of these three formalisms.
1.4 Practical Advantages of a Backbone Formalism

From a computational point of view, general models for which a clear parsing algorithm and a relatively fast implementation can be defined, produce fast and clear implementations for all particular instances. New research directions are also suggested by a general formalism. These directions are a consequence of instantiating the models’s parameters in a different way or by re-thinking the set of assumptions the particular instances have made. A brief description of the directions explored in this thesis follows.

Explicit use of probabilistic automata: Earlier, we mentioned that Markov models are heavily used in parsing models and that they can be replaced by probabilistic regular languages. Since our formalism is not bound to Markov models, we can use any algorithm for inducing probabilistic automata. In Chapter 5, we explore this idea. We define a type of grammar that uses probabilistic automata for building the set of rules. We compare two different classes of grammars depending on the algorithm used for learning the probabilistic automata. One of them is based on $n$-grams, and the other one is based on the minimum divergence algorithm (MDI). We show that the MDI algorithm produces both smaller and better performing grammars.

Splitting the training material: The fact that probabilistic automata replace Markov chains in the definition of our model allows us to think of a regular language as the union of smaller, more specific sublanguages. Our intuition is that the sublanguages are easier to induce and that the combination of them fully determines the whole language. In Chapter 6, we explore this idea by splitting the training material before inducing the probabilistic automata, then inducing one automaton for each component, and, finally, combining them into one grammar. We show that in this way, a measure that correlates well with parsing performance can be defined over grammars.

Sequences as features: Our formalism allows us to isolate particular aspects of parsing. For example, the linear order in which arguments appear in a parse tree is a fundamental feature used by language models. In Chapter 7, we investigate which sequences of information better predict sequences of dependents. We compare sequences of part-of-speech tags to sequences of non-terminal labels. We show that part-of-speech tags are better predictors of dependents.
1.5 What Can You Find in this Thesis?

In my opinion there are two different types of research. The first one pushes the frontier of knowledge forward, jumping from one point to a more advanced, better performing one. This pushing forward is sometimes carried out in a disorderly way, leaving many gaps along the way. The second line of research tries to fill in these gaps. Both types are very important, and neither of them can exist without the other. The second provides a solid foundation to the first one in order to make new jumps possible. After a jump, a huge amount of work is waiting to be done by the second type of research.

This thesis belongs solidly to the second type of research. Here, the reader will find a formal analysis of existing models. The reader will also find a general model that encompasses many of the models studied, as well as some properties these models enjoy — properties that we want the models to have and properties that were not known before and that the general model lets us prove. Finally, the reader will find a few explorations along new research directions also suggested by our model.

We hope that after having read the thesis, the reader will understand the language modeling task better. We also hope to have provided the area of natural language modeling with a more solid background. This background comprises consistency and expressive properties generally believed but not formally proven. In the thesis we also provide initial steps in promising new research directions.

In contrast, the reader will not find a state-of-the-art language model here. The reader will not find any claims regarding the universal structure natural language possesses either. It is very clear to me that the structure of natural language is, at this point, as unknown as it was when I first started. I can only see, so far, that formal languages with complexity up to context freeness can help us quite a lot in handling most syntactic structures.

1.6 Thesis Outline

Chapter 2 (Background and Landscape): This chapter introduces the machinery of formal languages. It covers formal language theory from regular languages to machine learning-based parsing algorithms, touching on context free grammars, W-grammars, and other formalisms.

Chapter 3 (The Role of Probabilities): This chapter investigates the role of probabilities in probabilistic context free grammars. Among others it answers questions like: “can probabilities be mimicked with rules?”, “can a grammar fully disambiguate a language?"
Chapter 4 (CW-Grammars as a General Model): This chapter presents our formalism. It shows that bilexical grammars, Markovian context free grammars and stochastic tree substitution grammars are instances of our formalism.

Chapter 5 (Alternative Approaches for Generating Bodies of Grammar Rules): This chapter explores the replacement of n-gram models by a more general algorithm for inducing probabilistic automata. It shows that the alternative algorithm produces smaller and better performing grammars.

Chapter 6 (Splitting training material optimally): This chapter investigates different ways to split the training material before it is used for inducing probabilistic automata. It defines a measure over grammars that correctly predicts their parsing performance.

Chapter 7 (Sequences as Features): This chapter focuses on a very specific aspect of syntax. We compare how two different features, each of them based on sequences of information, help to predict dependents of verbs. One of the features is based on sequences of part-of-speech tags while the other is based on sequences of non-terminals labels.

Chapter 8 (Conclusions): This chapter summarizes and combines conclusions of all the chapters.

Appendix A (Parser Implementation): This appendix discusses aspects related to the implementation of our parsing algorithm for probabilistic CW-grammars. It discusses the prerequisites a grammar has to fulfill for a parser to return the most probable tree. It also discusses some of the optimization techniques we implemented to reduce parsing time.

Appendix B (STOP symbol): This appendix reviews Collins's (1999) explanation of the necessity of the STOP symbol. The appendix uses Markov chains theory to re-explain and justify its necessity.