MAXIMUM ENTROPY MODELS FOR NATURAL LANGUAGE AMBIGUITY RESOLUTION

Adwait Ratnaparkhi

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1998

Professor Mitch Marcus
Supervisor of Dissertation

Professor Mark Steedman
Graduate Group Chair

Acknowledgements

The best aspect of a research environment, in my opinion, is the abundance of bright people with whom you argue, discuss, and nurture your ideas. I thank all of the people at Penn and elsewhere who have given me the feedback that has helped me to separate the good ideas from the bad ideas. I hope that I have kept the good ideas in this thesis, and left the bad ideas out!

I would like to acknowledge the following people for their contribution to my education:

- I thank my advisor Mitch Marcus, who gave me the intellectual freedom to pursue what I believed to be the best way to approach natural language processing, and also gave me direction when necessary. I also thank Mitch for many fascinating conversations, both personal and professional, over the last four years at Penn.

- I thank all of my thesis committee members: John Lafferty from Carnegie Mellon University, Aravind Joshi, Lyle Ungar, and Mark Liberman, for their extremely valuable suggestions and comments about my thesis research.

- I thank Mike Collins, Jason Eisner, and Dan Melamed, with whom I've had many stimulating and impromptu discussions in the LINC lab. I owe them much gratitude for their valuable feedback on numerous rough drafts of papers and thesis chapters.

- I thank Breck Baldwin, Mickey Chandrashekhar, Paola Merlo, Tom Morton, Martha Palmer, Jeff Reynar, Anoop Sarkar, Bangalore Srinivas, and others at IRCS, with whom I've had many interesting discussions about natural language that extended well beyond the scope of my thesis. I especially thank Jeff Reynar for his pivotal role in the work that comprises Chapter 4 of this thesis.

- I thank my past and current officemates: Kyle Hart, Karin Kipper, Dimitris Samaras, William Schuler, and Fei Xia, for many enjoyable discussions.

- I owe a great debt to Salim Roukos, Todd Ward, and the other members of the Speech Recognition Group at the IBM TJ Watson Research Center, for introducing me to maximum entropy modeling technology, and also to statistical natural language processing in general. My employment there as a programmer (1992—1994) was a very satisfying intellectual experience.

- Most of all, I thank my family for their role in my education. I thank my parents, and my brother Sachin, for the sacrifices they made to finance my undergraduate education at Princeton. I also thank my parents for teaching me the value of education at a young age. I thank my wife Amy for her steady support, encouragement, and love through the difficult times in my graduate career. I owe all of my success to the essential things that my family has given me over the years.

Abstract

MAXIMUM ENTROPY MODELS FOR NATURAL LANGUAGE AMBIGUITY RESOLUTION

Adwait Ratnaparkhi

Supervisor: Professor Mitch Marcus

This thesis demonstrates that several important kinds of natural language ambiguities can be resolved to state-of-the-art accuracies using a single statistical modeling technique based on the principle of maximum entropy.

We discuss the problems of sentence boundary detection, part-of-speech tagging, prepositional phrase attachment, natural language parsing, and text categorization under the maximum entropy framework. In practice, we have found that maximum entropy models offer the following advantages:

State-of-the-art Accuracy: The probability models for all of the tasks discussed perform at or near state-of-the-art accuracies, or outperform competing learning algorithms when trained and tested under similar conditions. Methods which outperform those presented here require much more supervision in the form of additional human involvement or additional supporting resources.

Knowledge-Poor Features: The facts used to model the data, or *features*, are linguistically very simple, or "knowledge-poor", but yet succeed in approximating complex linguistic relationships.

Reusable Software Technology: The mathematics of the maximum entropy framework

are essentially independent of any particular task, and a single software implementation can be used for all of the probability models in this thesis.

The experiments in this thesis suggest that experimenters can obtain state-of-the-art accuracies on a wide range of natural language tasks, with little task-specific effort, by using maximum entropy probability models.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis demonstrates that a single implementation of a statistical modeling technique based on the principle of maximum entropy, in conjunction with knowledge-poor information sources, suffices to achieve state-of-the-art performance in several tasks of tremendous interest to the natural language processing community.

Specifically, the thesis discusses the tasks of sentence boundary detection, part-of-speech tagging, prepositional phrase attachment, parsing, and text categorization. Here are some examples:

Sentence Boundary Detection: In the following text fragment,

`He called Mr. White at 4 p.m. in Washington, D.C. Mr. Green responded.`

how can a computer program tell which of the .'s, if any, denote *actual* sentence boundaries ?

Part-of-Speech Tagging: In the following two sentences,

- Fruit flies like a banana.
- Time flies like an arrow.

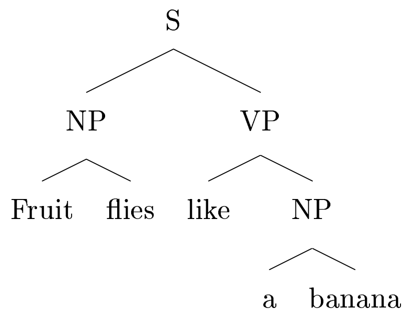
the words *flies* and *like* are ambiguous. In the first sentence, *flies* is a noun and *like* is a verb, while in the second sentence, *flies* is a verb and *like* is a preposition. How can a computer program automatically and accurately predict the part-of-speech of ambiguous words like *flies* and *like* ?

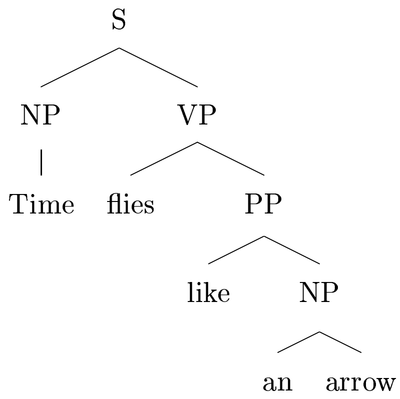
Prepositional Phrase Attachment: In the following two sentences,

- He bought the car with a credit card.
- He bought the car with a sunroof.

what does a computer program need to know in order to know that *with a credit card* refers to *bought*, whereas *with a sunroof* refers to *the car* ?

Parsing: A natural language parser takes a sentence as input, and determines the labelled syntactic tree structure that corresponds to the interpretation of the sentence. For example, the different part-of-speech assignments for the word *flies* and *likes* lead to different parse trees, and different interpretations:

- 

- 

Parsing requires the resolution of *all* syntactic ambiguities that arise during the interpretation of a sentence, and not just the noun/verb ambiguities or prepositional phrase attachments discussed earlier. How can a computer automatically predict all the plausible tree structures, and then choose among them to resolve any structural ambiguities ?

Text Categorization: Given a document and a topic, the task is to decide if the document should be categorized with the topic. If the document contains the word *money*, is it relevant to the topic of *mergers and acquisitions* ? How can the computer best use the words in the documents to predict the topic ?

Each task can be viewed as a "classification" problem, in which the objective is to estimate a function $cl : X \rightarrow Y$, which maps an object $x \in X$ to its "correct" class $y \in Y$. Typically, Y is the predefined set of linguistic classes we are interested in predicting, and X consists of either words, sentences, or other textual material of interest that might be useful for making the prediction. For example, in sentence boundary detection, given a potential end-of-sentence mark $x \in \{. \ ! \ ?\}$, we wish to predict $y \in \{true, false\}$ which classifies it as either a real or spurious sentence boundary. In POS tagging, given an n word input sentence $x \in \{\text{all possible } n \text{ word sentences}\}$, we wish to predict a sequence of n tags $y \in T^n$, where T are the allowable POS tags for a word. For complex problems like tagging and parsing, it is computationally convenient to decompose them into a *sequence* of simpler classification problems. For example, instead of building a classifier to predict a sequence of n tags, it is simpler to first estimate a classifier that predicts a POS tag for a single word, and to then apply it n times, once for each word. Likewise, instead of predicting a whole parse tree, we predict a sequence of simpler actions, one at a time, that each predict a small part of the parse tree. Since the simpler classes are predicted in sequence, a classifier can exploit the previously completed $n-1$ classifications in order to correctly predict the nth class in the sequence. The exact details of decomposing a given task are a problem-specific art, but the general methodology is applicable to any complex linguistic prediction task.

All the classification functions for the tasks discussed in this thesis are implemented with maximum entropy probability models. We can implement any classifier $cl : X \rightarrow Y$ with a conditional probability model p by simply choosing the class with the highest conditional probability:

$$cl(x) = \arg\max_{y} p(y|x)$$

where x is an textual object and y is a class. Likewise, probability models can naturally implement a complex classifier $cl : X^n \rightarrow Y^n$ as a sequence of simpler probability calculations:

$$cl(x_1 \ldots x_n) = \arg\max_{y_1 \ldots y_n} \prod_{i=1}^{n} p(y_i | x_1 \ldots x_n, y_1 \ldots y_{i-1})$$

where $x_1 \ldots x_n, y_1 \ldots y_{i-1}$, informally called a *context* or *history*, is the textual material available at the ith decision, and where y_i is the outcome of the ith decision. Under the maximum entropy framework, the probability for a class y and object x depends solely on the *features* that are "active" for the pair (x, y), where a feature is defined here as a function $f : (X, Y) \rightarrow \{0, 1\}$ that maps a pair (x, y) to either 0 or 1. Features are the means through which an experimenter feeds problem-specific information to the maximum entropy model, and they should encode any information that could be useful in correctly determining the class. The importance of each feature is determined automatically by running a parameter estimation algorithm over a pre-classified set of examples, or a "training set". As a result, an experimenter need only tell the model *what* information to use, since the model will automatically determine *how* to use it.

This thesis will provide experimental support for three claims, regarding *accuracy*, *knowledge-poor features*, and *reusability*:

Accuracy In every application of maximum entropy modelling discussed here, the accuracy is at or near the state-of-the-art, even though we have not tuned the models in any substantial task-specific manner. The few published results that exceed those presented here require considerably more domain expertise or human effort on the part of the experimenter. In controlled experiments, our maximum entropy model implementation outperformed a commercially available decision-tree learning package.

Knowledge-Poor Features While the primary objective in designing a feature set is to maximize prediction accuracy, the feature sets in this thesis are comparatively *knowledge-poor*, in that they do not require deep linguistic knowledge, and ask only elementary questions about the surrounding context. The feature sets used in this thesis rely less on linguistic knowledge, preprocessing, or semantic databases than competing approaches, and are therefore much easier to specify and easier to port than the features used in these other approaches. Despite the apparent simplicity of the features, they can effectively approximate complex linguistic relationships, particularly in the case of parsing and prepositional attachment tasks.

Software Reusability The generality of the maximum entropy framework allows an experimenter to use literally the same parameter estimation routine for different tasks. The code for the parameter estimation is essentially *independent* of any particular task, and a single implementation suffices for all the models in this thesis. More importantly, the maximum entropy models perform reasonably well on each task, despite the fact that all the tasks are quite different in nature and complexity. The experimental results in this thesis suggest that researchers can use and re-use a single implementation of the maximum entropy framework for a wide variety of tasks, and expect it to perform at state-of-the-art accuracies.

Chapter 2 describes the maximum entropy framework, Chapter 3 discusses other learning techniques for natural language processing, and Chapters 4, 5, 6, 7 discuss the tasks of sentence boundary detection, part-of-speech tagging, parsing, and prepositional phrase attachment, respectively. Chapter 8 discusses comparative experiments with other feature selection and learning techniques, Chapter 9 discusses some drawbacks of the technique, and Chapter 10 discusses our conclusions.

Chapter 2

The Maximum Entropy Framework

2.1 Introduction

As noted in the previous chapter, many problems in natural language processing (NLP) can be re-formulated as classification problems, in which the task is to observe some linguistic "context" $b \in \mathcal{B}$ and predict the correct linguistic "class" $a \in \mathcal{A}$. This involves constructing a classifier $cl : \mathcal{B} \rightarrow \mathcal{A}$, which in turn can be implemented with a conditional probability distribution p, such that $p(a|b)$ is the probability of "class" a given some "context" b. Contexts in NLP tasks usually include at least words, and the exact context depends on the nature of the task; for some tasks, the context b may consist of just a single word, while for others, b may consist of several words and their associated syntactic labels. Large text corpora usually contain some information about the cooccurrence of a's and b's, but never enough to reliably specify $p(a|b)$ for all possible (a, b) pairs, since the words in b are typically sparse. The challenge is then to find a method for using the *partial evidence* about the a's and b's to reliably estimate the probability model p.

Maximum entropy probability models offer a clean way to combine diverse pieces of contextual evidence in order to estimate the probability of a certain linguistic class occurring with a certain linguistic context. We first demonstrate how to represent evidence and combine it with a particular form of probability model in the maximum likelihood framework, and then discuss an independently motivated interpretation of the probability model under the maximum entropy framework. We describe the framework first as it applies to an example problem, and then as it applies to the NLP problems in this thesis. We also discuss the advantages of combining evidence in this framework.

2.2 Representing Evidence

We represent evidence with functions known as *contextual predicates* and *features*. If $\mathcal{A} = \{a_1 \ldots a_q\}$ represents the set of possible classes we are interested in predicting, and $\mathcal{B}$ represents the set of possible contexts or textual material that we can observe, then a contextual predicate is a function:

$$cp : \mathcal{B} \rightarrow \{true, false\}$$

that returns $true$ or $false$, corresponding to the presence or absence of useful information in some context, or *history* $b \in \mathcal{B}$. The exact set of contextual predicates $cp_1 \ldots cp_m$ that are available for use varies with the problem, but in each problem they must be supplied by the experimenter. Contextual predicates are used in features, which are functions of the form

$$f : \mathcal{A} \times \mathcal{B} \rightarrow \{0, 1\}$$

Any feature[1] in this thesis has the form

$$f_{cp,a'}(a,b) = \begin{cases} 1 & \text{if } a = a' \text{ and } cp(b) = true \\ 0 & \text{otherwise} \end{cases}$$

and checks for the co-occurrence of some prediction a' with some contextual predicate cp. The actual set of features we use for a problem is determined by a *feature selection* strategy, which, in general, is specific to the problem. We will later show that a single feature selection strategy applied to the different problems in this thesis still yields good prediction accuracy.

[1]While a *feature* in this thesis is defined on $\mathcal{A} \times \mathcal{B}$, a *feature* in the machine learning literature is usually defined *only* on the space of possible contexts $\mathcal{B}$. Our definition of *feature* is borrowed from past literature on the maximum entropy framework.

2.3 The Machine Learning or Corpus-Based Approach

The work in this thesis fits squarely in what is called the "machine learning" or "corpus-based" approach to natural language processing. In this approach, we assume the existence of a *training set* $\mathcal{T} = \{(a_1, b_1) \ldots (a_N, b_N)\}$, which is a large set of contexts $b_1 \ldots b_N$ that have been annotated with their correct classes $a_1 \ldots a_N$. The notion of a training set that consists of pairs of boolean vectors (contexts) together with classes is very general, and is used by a vast number of algorithms in the machine learning literature. The advantage of conforming to this representation is that experimenters can use the learning technique of their choice, and that rigorous comparisons can be made between different learning techniques on the same data.

2.3.1 Learning with Maximum Likelihood Estimation on Exponential Models

One way to combine evidence is to "weight" the features by using them in a log-linear, or exponential, model:

$$\begin{aligned} p(a|b) &= \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \\ Z(b) &= \sum_a \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \end{aligned} \tag{2.1}$$

where k is the number of features and $Z(b)$ is a normalization factor to ensure that $\sum_a p(a|b) = 1$. Each parameter α_j, where $\alpha_j > 0$, corresponds to one feature f_j and can be interpreted as a "weight" for that feature. The probability $p(a|b)$ is then a normalized product of those features that are "active" on the (a, b) pair, i.e., those features f_j such that $f_j(a, b) = 1$. The weights $\alpha_1 \ldots \alpha_k$ of the probability distribution p^* that best fit the training data can be obtained with the popular technique of maximum likelihood estimation:

$$\begin{aligned} Q &= \{p \mid p(a|b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}\} \\ L(p) &= \sum_{a,b} \tilde{p}(a,b) \log p(a|b) \\ p^* &= \arg\max_{q \in Q} L(q) \end{aligned}$$

where Q is the set of models of log-linear form, $\tilde{p}(a, b)$ is the probability of seeing a, b in the training set $\mathcal{T}$, $L(p)$ is the conditional log-likelihood of the training set $\mathcal{T}$ (normalized by the number of training events), and p^* is the optimal probability distribution according to the maximum likelihood criterion.

2.3.2 Learning under the Maximum Entropy Framework

While there are conceivably many other ways to combine evidence in the form of a probability distribution, the form (2.1) has an independently motivated justification under the maximum entropy framework. The *Principle of Maximum Entropy* [Jaynes, 1957, Good, 1963], argues that the best probability model for the data is the one which maximizes entropy, over the set of probability distributions that are consistent with the evidence.

> ...in making inferences on the basis of partial information we must use that probability distribution which has maximum entropy subject to whatever is known. This is the only unbiased assignment we can make; to use any other would amount to arbitrary assumption of information which by hypothesis we do not have.

We will first illustrate maximum entropy modeling with a simple example, and then describe how the framework is applied to the natural language ambiguity problems in this thesis.

2.4 Maximum Entropy: A simple example

The following example illustrates the use of maximum entropy on a very simple problem. Suppose the task is to estimate a joint probability distribution p defined over $\{x, y\} \times \{0, 1\}$. Furthermore suppose that the only facts known about p are that $p(x, 0) + p(y, 0) = .6$, and that $p(x, 0) + p(y, 0) + p(x, 1) + p(y, 1) = 1.0$. (While the constraint that $\sum_{a,b} p(a, b) = 1$ is implicit since p is a probability distribution, it will be treated as an externally imposed constraint for illustration purposes.)

In a prediction task, x and y would be mutually exclusive observations, and 0 and 1 would be two mutually exclusive outcomes we are interested in predicting. For example,

$p(a,b)$	0	1	
x	?	?	
y	?	?	
total	.6		1.0

Table 2.1: Task is to find a probability distribution p under constraints $p(x,0)+p(y,0) = .6$, and $p(x,0) + p(x,1) + p(y,0) + p(y,1) = 1$

suppose the actual task is to determine the probability with which first-year students do not receive "A" grades, and suppose we assign the following interpretation to the event space $\{x,y\} \times \{0,1\}$:

$$\begin{aligned} x &= \text{student is a first-year} \\ y &= \text{student is not a first-year} \\ 0 &= \text{student's grade is A} \\ 1 &= \text{student's grade is not A} \end{aligned}$$

Then the observed fact that "60% of all students received an A for a grade" is implemented with the constraint $p(x,0) + p(y,0) = .6$. The implicit fact that "100% of all students received an A or not an A" is implemented with the constraint $\sum_{a,b} p(a,b) = 1$. The goal in this modeling framework is to fully estimate p, so questions such as "What (estimated) percentage of first-year students did not receive an A?" can be answered by computing a probability, such as $p(x,1)$.

Table 2.1 represents the probability distribution p as 4 cells labelled with "?", whose values must be consistent with the constraints. Clearly there are (infinitely) many consistent ways to fill in the cells of table 2.1; one such way is shown in table 2.2. However, the Principle of Maximum Entropy recommends the assignment in table 2.3, which is the most non-committal assignment of probabilities that meets the constraints on p.

Formally, under the maximum entropy framework, the fact

$$p(x,0) + p(y,0) = .6$$

	0	1	
x	.5	.1	
y	.1	.3	
total	.6		1.0

Table 2.2: One way to satisfy constraints

	0	1	
x	.3	.2	
y	.3	.2	
total	.6		1.0

Table 2.3: The most "uncertain" way to satisfy constraints

is implemented as a constraint on the model p's expectation of the feature f_1:

$$E_p f_1 = .6 \tag{2.2}$$

where

$$E_p f_1 = \sum_{a \in \{x,y\}, b \in \{0,1\}} p(a,b) f_1(a,b)$$

and where f_1 is defined as follows:

$$f_1(a,b) = \begin{cases} 1 & \text{if } b = 0 \\ 0 & \text{otherwise} \end{cases}$$

Similarly, the fact

$$p(x,0) + p(y,0) + p(x,1) + p(y,1) = 1.0$$

is implemented as the constraint

$$E_p f_2 = 1.0 \tag{2.3}$$

where

$$\begin{aligned} E_p f_2 &= \sum_{a \in \{x,y\}, b \in \{0,1\}} p(a,b) f_2(a,b) \\ f_2(a,b) &= 1 \end{aligned}$$

The objective under the maximum entropy framework is then to maximize

$$H(p) = -\sum_{a\in\{x,y\},b\in\{0,1\}} p(a,b)\log p(a,b)$$

subject to the constraints (2.2) and (2.3).

Assuming that features always map an event (a,b) to either 0 or 1, a constraint on a feature expectation is simply a constraint on the sum of the (a,b) cells in the table on which the feature returns 1. While the above constrained maximum entropy problem can be solved trivially (by inspection), an iterative procedure is usually required for larger problems since multiple constraints often overlap in ways that prohibit a closed form solution.

2.5 Conditional Maximum Entropy Models

While the previous example used only two features, the framework used to solve the problems in this thesis assumes that we have k features, and that given a linguistic prediction $a \in \mathcal{A}$ and an observable context $b \in \mathcal{B}$, our ultimate goal is to find an estimate for the conditional probability $p(a|b)$, as opposed to the joint probability. In the conditional maximum entropy framework used in earlier work such as [Berger et al., 1996, Lau et al., 1993, Rosenfeld, 1996], the optimal solution p^* is the most uncertain distribution that satisfies the k constraints on feature expectations:

$$\begin{aligned}
p^* &= \arg\max_{p\in P} H(p) \\
H(p) &= -\sum_{a,b} \tilde{p}(b)p(a|b)\log p(a|b) \\
P &= \{p \mid E_p f_j = E_{\tilde{p}} f_j,\ j = \{1\ldots k\}\} \\
E_{\tilde{p}} f_j &= \sum_{a,b} \tilde{p}(a,b) f_j(a,b) \\
E_p f_j &= \sum_{a,b} \tilde{p}(b)p(a|b) f_j(a,b)
\end{aligned}$$

An important difference here from the simple example is that $H(p)$ denotes the *conditional* entropy averaged over the training set, as opposed to the joint entropy, and that the marginal probability of b used here is the observed probability $\tilde{p}(b)$, as opposed to a model probability $p(b)$. Our choice of $\tilde{p}(b)$ as a marginal probability, borrowed from earlier work such as [Berger et al., 1996], is motivated by the fact that any model probability $p(b)$ cannot be explicitly normalized over the space of possible contexts $\mathcal{B}$, since $\mathcal{B}$ is typically very large in practice. Here $E_p f_j$ is the model p's expectation of f_j, and is computed differently than before since it uses $\tilde{p}(b)$ as a marginal probability, for the same important practical reasons. As before, $E_{\tilde{p}} f_j$ denotes the observed expectation of a feature f_j, $\tilde{p}(a, b)$ denotes the observed probability of (a, b) in some fixed training sample, and P denotes the set of probability models that are consistent with the observed evidence.

2.5.1 Relationship to Maximum Likelihood

In general, the maximum likelihood and maximum entropy frameworks are two different approaches to statistical modeling, but in this case they yield the same answer. We can show that maximum likelihood parameter estimation for models of form (2.1) is equivalent to maximum entropy parameter estimation over the set of consistent models. That is,

$$p^* = \arg\max_{q \in Q} L(q) = \arg\max_{p \in P} H(p)$$

This fact is described using lagrange multiplier theory in [Berger et al., 1996], and with information theoretic arguments (for the case when p^* is a joint model) in [Della Pietra et al., 1997]. Under the maximum likelihood criterion, p^* will fit the data as closely as possible, while under the maximum entropy criterion, p^* will not assume anything beyond the information in the linear constraints that define P. We include a proof in Section A.2 to show that the condition $p^* = \arg\max_{q \in Q} L(q)$ is equivalent to the condition that $p^* = \arg\max_{p \in P} H(p)$. It is important to note that the model form (2.1) is not arbitrary, the maximum entropy solution $p^* = \arg\max_{p \in P} H(p)$ *must* have this form. This duality with the maximum entropy principle is appealing, and provides both an interpretation and a justification for using maximum likelihood estimation on models of form (2.1).

2.6 Parameter Estimation

We use an algorithm called *Generalized Iterative Scaling*[Darroch and Ratcliff, 1972], or GIS, to find values for the parameters of p^*. The GIS procedure requires that the features sum to a constant for any $(a,b) \in \mathcal{A} \times \mathcal{B}$, or that

$$\sum_{j=1}^{k} f_j(a,b) = C \tag{2.4}$$

where C is some constant. If this condition is not already true, we use the training set to choose C

$$C = \max_{a \in \mathcal{A}, b \in \mathcal{T}} \sum_{j=1}^{k} f_j(a,b)$$

and add a "correction" feature f_l, where $l = k+1$, such that

$$f_l(a,b) = C - \sum_{j=1}^{k} f_j(a,b)$$

for any (a,b) pair. Note that unlike the existing features, $f_l(a,b)$ ranges from 0 to C, where C can be greater than 1. In theory, a correction constant to enforce the constraint (2.4) for all (a,b) pairs should be derived from the space of possible events $\mathcal{A} \times \mathcal{B}$. However, a summation over the whole event space is not practical, and correction constants derived from training sets are usually accurate in practice, if the training set is large.

Algorithm 1 (Generalized Iterative Scaling (GIS)). *The following procedure will converge to p^*:*

$$\begin{aligned} \alpha_j^{(0)} &= 1 \\ \alpha_j^{(n+1)} &= \alpha_j^{(n)} [\frac{E_{\tilde{p}} f_j}{E_{p^{(n)}} f_j}]^{\frac{1}{C}} \end{aligned} \tag{2.5}$$

where

$$\begin{aligned} E_{p^{(n)}} f_j &= \sum_{a,b} \tilde{p}(b) p^{(n)}(a|b) f_j(a,b) \\ p^{(n)}(a|b) &= \frac{1}{Z(b)} \prod_{j=1}^{l} (\alpha_j^{(n)})^{f_j(a,b)} \end{aligned}$$

[Darroch and Ratcliff, 1972] shows[2] that $L(p^{(n+1)}) \geq L(p^{(n)})$, and that $\lim_{n\to\infty} p^{(n)} = p^*$. See [Csiszar, 1989] for a proof of GIS under the I-divergence geometry framework of [Csiszar, 1975]. GIS is actually a special case of *Improved Iterative Scaling*, described in [Berger et al., 1996, Della Pietra et al., 1997], which finds the parameters of p^* without the use of a correction feature.

2.6.1 Computation

Given k features, the GIS procedure requires computation of each observed expectation $E_{\tilde{p}} f_j$, and requires re-computation of the model's expectation $E_p f_j$ on each iteration, for $j = 1 \ldots k$. The quantity $E_{\tilde{p}} f_j$ is merely the count of f_j normalized over the training set:

$$E_{\tilde{p}} f_j = \sum_{a,b} \tilde{p}(a,b) f_j(a,b) = \frac{1}{N} \sum_{i=1}^{N} f_j(a_i, b_i)$$

where N is the number of event *tokens* (as opposed to *types*) the training sample $\mathcal{T} = \{(a_1, b_1), \ldots, (a_N, b_N)\}$,

The computation of $E_p f_j$ involves summing over each context b in the training set, and each $a \in \mathcal{A}$:

$$E_{p^{(n)}} f_j = \sum_{a,b} \tilde{p}(b) p^{(n)}(a|b) f_j(a,b)$$

Most importantly, any context b not in the training set can be excluded from this sum, since $\tilde{p}(b) = 0$ if $b \notin \mathcal{T}$. The computation of $E_p f_j$ dominates the running time of each iteration. If N is the number of training samples, $\mathcal{A}$ is the set of predictions, and V is the average number of features that are active for a given event, the running time of each iteration is $O(N|\mathcal{A}|V)$.

The procedure should be terminated after a fixed number of iterations or when the change in log-likelihood or accuracy is negligible. For the problems in this thesis, using 100 iterations is a good "rule-of-thumb", since using more iterations rarely resulted in any significant accuracy gains.

[2]The proof in [Darroch and Ratcliff, 1972] is for the case when p^* is a joint model, the proof for when p^* is a conditional model is similar.

2.7 Discussion

The biggest advantage of this framework is that it allows a virtually unrestricted ability to represent problem-specific knowledge in the form of features. The exact linguistic information corresponding to a feature is dependent on the task, but there is no inherent restriction on what kinds of linguistic items a feature can encode. The features in a maximum entropy model need not be statistically independent; all probability models in this thesis fully exploit this advantage by using overlapping and interdependent features. In tasks which require a sequence of classification decisions, like tagging and parsing, it is highly likely that the features of the model used for the nth decision in the sequence will look at one or more of the $n-1$ previous classification decisions. For example, [Ratnaparkhi, 1996] estimates a model for part-of-speech tagging in which the context b contains the word to be tagged, surrounding words, as well as the results of the previous two tagging decisions (i.e., the tags of the previous two words). For example, useful features for part-of-speech tagging might be

$$f_j(a,b) = \begin{cases} 1 & \text{if } a = \texttt{DETERMINER} \text{ and } \texttt{current\_word\_is\_``that''}(\text{b}) = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

or

$$f_k(a,b) = \begin{cases} 1 & \text{if } a = \texttt{NOUN} \text{ and } \texttt{previous\_tag\_is\_DETERMINER}(\text{b}) = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

The observed pieces of evidence corresponding to these features are $E_{\tilde{p}} f_j$ and $E_{\tilde{p}} f_k$, which have clear intuitive interpretations. $E_{\tilde{p}} f_j$ is the frequency in the training sample with which "that" occurs as a DETERMINER normalized over the number of training samples, while $E_{\tilde{p}} f_k$ is the frequency in the training sample with which DETERMINER precedes NOUN, also normalized over the number of training samples.

In the maximum entropy framework, experimenters can add, without modifying the formalism, more exotic and detailed forms of evidence once they are discovered. As an example, a more interesting feature might be

$$f_j(a,b) = \begin{cases} 1 & \text{if } a = \texttt{ADVERB} \text{ and } \texttt{complex\_``about''\_predicate}(\text{b}) = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

where

$$\texttt{complex\_“about”\_predicate}(b) = \begin{cases} true & \text{if current word in } b \text{ is “about”} \\ & \text{and next word is “\$” or a number} \\ false & \text{otherwise} \end{cases}$$

Such a feature would help a tagger to correctly distinguish the case where *about* is used as an adverb (like *approximately*), as in *It cost about $5.00* from the case where *about* is used as a preposition, as in *He talked about it.* While such features are diverse in nature, the extent to which each feature f_j contributes towards $p(a|b)$, i.e., its "weight" α_j, is automatically determined by the Generalized Iterative Scaling algorithm, described in Section 2.6. As a result, experimenters need only focus their efforts on discovering what features to use, and not on how to use them.

2.7.1 A Special Case: Non-Overlapping Features

The maximum entropy framework reduces to a very simple type of probability model when the features do not overlap. Suppose that the contextual predicates partition $\mathcal{B}$, so that every b corresponds to only one predicate cp_i. i.e., $cp_i(b) = true$ while $cp_{i'}(b) = false$ for $i \neq i'$. Furthermore suppose that for every predicate cp_i, we have $|\mathcal{A}|$ features $f_{cp_i,a}$ which test for cp_i and a. In this case, an iterative algorithm is not necessary to compute $p(a|b)$, the closed form estimate is simply a ratio of counts:

$$p(a|b) = \frac{E_{\tilde{p}} f_{cp,a}}{E_{\tilde{p}}[cp(b) = true]} = \frac{Count(cp(b) = true, a)}{Count(cp(b) = true)} \tag{2.6}$$

where cp is the predicate that corresponds to b. (See Section A.1 for a proof.) So if the features form partitions of the event space as described above, the parameter estimation algorithm, and the maximum entropy framework itself, are not useful, since the correct probability estimate can be derived from the raw counts alone. We emphasize that true utility of the maximum entropy framework comes from its ability to robustly combine features that do not form partitions of the event space, but instead overlap in arbitrarily complex ways.

2.8 Conclusion

The rest of this thesis will demonstrate that the maximum entropy framework discussed here is general enough to handle, without modification, a wide range of natural language problems. The only items in the framework that are particular to each task are the set of outcomes $\mathcal{A}$, the set of possible contexts $\mathcal{B}$, and the set of features $f_1 \ldots f_k$ in the model. All three items are fully specified with these three facts:

Outcomes: The set of possible predictions $\mathcal{A} = \{a_1, \ldots, a_q\}$.

Contextual Predicates: The available contextual predicates $\{cp_1, \ldots, cp_m\}$; these are necessary and sufficient to capture all the information in any context $b \in \mathcal{B}$.

Feature Selection: The features actually used in the model. A particular contextual predicate cp_i may occur with many predictions but may be useful for predicting only a few of them. Even if cp_i occurs with both $a_j \in \mathcal{A}$ and $a_k \in \mathcal{A}$, the model may use the feature f_{cp_i,a_j}, but not the feature f_{cp_i,a_k},

Therefore, in subsequent chapters, a maximum entropy model will be described by only the above three characteristics, since all its other formal properties, namely:

- The form of the model
- The form of the constraints
- Its maximum entropy property
- Its relationship to maximum likelihood estimation
- The parameter estimation algorithm

are independent of the particular prediction task.

Chapter 3

Machine Learning Techniques Applied to Natural Language: A Brief Review

We illustrate some advantages and disadvantages of the maximum entropy framework by comparing it to other machine learning algorithms that have been applied to natural language. There are many statistical and corpus-based algorithms in the literature for natural language learning, but we restrict our discussion to those that are *general*, i.e., those that have not been specifically designed for one particular domain or application. In this discussion, we will assume the existence of a training set $\mathcal{T} = \{(a_1, b_1) \ldots (a_N, b_N)\}$ and m contextual predicates $cp_1 \ldots cp_m$. We also assume that the machine learning techniques discussed here use the training set to gather co-occurrence statistics between some outcome a and the truth value of any contextual predicate cp_i applied to a context b, or $cp_i(b)$. We review the other natural language learning techniques, and motivate our use of the maximum entropy framework.

3.1 Naive Bayes

The naive Bayes classifier is derived from Bayes' rule, and from strong conditional independence assumptions about the observed evidence. It has been used for natural language applications such as text categorization [Lewis and Ringuette, 1994] and word sense disambiguation [Gale et al., 1992]. Using Bayes' rule, we rewrite $p(a|b)$:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$

and use it to construct a classifier $cl_{bayes} : \mathcal{B} \rightarrow \mathcal{A}$:

$$cl_{bayes}(b) = \arg\max_{a} p(b|a)p(a)$$

Typically, the explicit computation of $p(b|a)$ is impossible due to sparse data, so most experimenters make a very strong conditional independence assumption:

$$p(b|a) = p(cp_1(b) \ldots cp_m(b)|a) = \prod_{i=1}^{m} p_i(cp_i(b)|a)$$

The parameters $p_i(cp_i(b)|a)$ are derived directly from the $(cp_i(b), a)$ counts in the training data and do not require an iterative estimation algorithm like the maximum entropy models.

The maximum entropy framework differs from the naive Bayes classifier in that it makes no inherent conditional independence assumptions, and allows experimenters to encode dependencies freely in the form of features, at the expense of an iterative parameter estimation algorithm. E.g., suppose cp_1 and cp_2 are contextual predicates whose results are independent, i.e., $p(cp_1(b) = X, cp_2(b) = Y) = p(cp_1(b) = X)p(cp_2(b) = Y)$ for any context b. Now define another predicate $cp_3(b) = cp_1(b) \wedge cp_2(b)$, which is clearly dependent on cp_1 and cp_2. In the maximum entropy framework, we can use features to correlate cp_1, cp_2, cp_3 with some prediction a', e.g.,

$$\begin{aligned}
f_1(a,b) &= \begin{cases} 1 & \text{if } a = a' \text{ and } cp_1(b) = true \\ 0 & \text{otherwise} \end{cases} \\
f_2(a,b) &= \begin{cases} 1 & \text{if } a = a' \text{ and } cp_2(b) = true \\ 0 & \text{otherwise} \end{cases} \\
f_3(a,b) &= \begin{cases} 1 & \text{if } a = a' \text{ and } cp_3(b) = true \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

without violating any independence assumptions. In contrast, the Naive Bayes probability model will treat cp_3 as if it were independent from cp_1, cp_2, and is therefore unlikely to yield accurate probability estimates. Under the maximum entropy framework, the parameter estimation algorithm is not explicitly told about any interdependencies expressed in the features, but yet will adjust the feature weights to account for them. Of course, there may be interdependencies that are *not expressed* in the features, in which case the parameter estimation algorithm will fail to account for them.

3.2 Statistical Decision Trees

In a statistical decision tree, or class probability tree, the internal nodes represent "tests" and the leaves represent conditional probability distributions. Any context $b \in \mathcal{B}$ corresponds to some leaf l of the decision tree. The path from the root to the leaf l is obtained by first applying the test at the current node, choosing a branch to a child node that corresponds to the outcome of the test, and then recursively repeating the process from the new child node. The conditional probability distribution associated with l, or p_l, is then used to compute $p(a|b)$.

We draw comparison to binary decision trees, where each internal node corresponds to a contextual predicate $cp : \mathcal{B} \to \{true, false\}$, where left branches correspond to $false$, and where right branches correspond to $true$. Assume that $b \in \mathcal{B}$ is a context, and that cp is the contextual predicate at the current node. We can trace a path from the root to a leaf if we start at the root, take the branch to the left child if $cp(b) = false$, take the branch to the right otherwise, and recursively repeat this process from the newly selected child node until we reach a leaf. A path from the root to the leaf l, corresponding to a context b, is then uniquely determined by the sequence of contextual predicates that are used in reaching l, and by the results of applying those predicates to b. Let cp_{l^i} denote the predicate that corresponds to the ith parent of leaf l, and let v_{l^i} denote the value of $cp_{l^i}(b)$, where b is a context associated with leaf l. Define $leaf_l : \mathcal{B} \to \{true, false\}$ to be a function that returns *true* if b corresponds to the unique leaf l:

$$leaf_l(b) = \begin{cases} true & \text{if } \bigwedge_{i=1}^{n} cp_{l^i}(b) = v_{l^i} \\ false & \text{otherwise} \end{cases}$$

The functions $leaf_l$, for all leaves l, partition $\mathcal{B}$ such that each partition corresponds to exactly one leaf. The conditional probability $p_l(a|b)$ at each leaf l (for any b such that $leaf_l(b) = true$) is simply the normalized frequency of a in the partition of $\mathcal{B}$ corresponding to the leaf l. I.e., it can be derived from the raw counts as follows:

$$p_l(a|b) = \frac{Count(leaf_l(b) = true, a)}{Count(leaf_l(b) = true)} \tag{3.1}$$

where $Count()$ returns the raw counts over the training set $\mathcal{T}$.

Statistical decision trees are similar to maximum entropy models in that they can cope with diverse, non-independent pieces of information in the predicates. We can even implement a decision tree with L leaves as a maximum entropy model with $L|\mathcal{A}|$ features, where $\mathcal{A}$ is the set of predictions. For each leaf l in the decision tree, the corresponding maximum entropy model has $|\mathcal{A}|$ features of the form:

$$f_{l,a'}(a,b) = \begin{cases} 1 & \text{if } leaf_l(b) = true \text{ and } a = a' \\ 0 & \text{otherwise} \end{cases}$$

When the predicates partition $\mathcal{B}$, and when the features check for all $|\mathcal{A}|$ outcomes with each predicate, the maximum entropy probability $p(a|b)$ can be derived simply from the raw counts in the training set using equation (2.6), and is identical to (3.1).

Statistical decision trees are grown from a training sample $\mathcal{T}$ with recursive partitioning algorithms, like those described used in the ID3[Quinlan, 1986], C4.5[Quinlan, 1992], and CART[Breiman et al., 1982]. These induction algorithms can automatically grow complicated tests (conjuncts of predicates) from simple tests (predicates). Our approach differs in that any conjunction of predicates are specified in advance by the experimenter in the form of features, and that complicated features are not derived automatically from simpler ones. However, feature induction is possible; see [Berger et al., 1996, Della Pietra et al., 1997] for an algorithm which incrementally grows conjuncts of features in the maximum entropy framework.

An important advantage of maximum entropy models over decision trees is that the maximum entropy parameter estimation does not partition the training sample $\mathcal{T}$. Partitioning the data with sparse predicates leads to uneven splits, which in turn lead to the well-known *data fragmentation* problem, in which the distributions at some leaves are unreliable since they correspond to a very small partition of the training set. Data fragmentation is particularly a concern for natural language processing, since predicates typically test for words, which are sparse by nature. Past work on decision trees for natural language, such as [Black et al., 1993, Jelinek et al., 1994, Magerman, 1995] has relied a host of other techniques to alleviate data fragmentation, such as clustering algorithms that reduce the amount (and hence sparseness) of predicates, as well as smoothing and pruning algorithms that yield better probability estimates. In contrast, the maximum entropy models used in this thesis do not use clustering and smoothing techniques.

3.3 Transformation Based Learning

Transformation based learning, introduced in [Brill, 1993a], is a non-probabilistic technique for incrementally learning rules to maximize prediction accuracy. A transformation rule, in our notation, would have the format

$$\text{If the outcome is } a\text{, and } cp_i(b) = true\text{, change } a \text{ to } a'$$

where a, a' are outcomes, and cp_i is a contextual predicate. Transformation based learning begins with an *initial state* T_0, which consists of all (a, b) pairs such that b is a context of the original training set and a is the default outcome for b, e.g., the most frequent outcome or the best guess for the context. Next, the learner iterates, and on the ith iteration, selects the transformational rule whose application to the (a, b) pairs in T_{i-1} results in the highest score. The (a, b) pairs of T_{i-1} are re-annotated with this selected rule to create T_i. The score of a transformational rule on the ith iteration is usually related to how much it improves the resemblance of T_{i-1} with respect to the truth, i.e., the manually annotated training set $\mathcal{T}$. The transformation based learning strategy can therefore be viewed as greedy error minimization. The experimenter is required to specify the initial state, the space of transformations available to the learner, and the scoring function. When given a context b in test data, we begin with the default outcome for b, and apply, in respective order, the transformations learned during the training phase. Past literature[Brill, 1994] on transformation based learning claims that the rules learned by the procedure are easier to understand than the statistics of comparable probabilistic approaches. Transformation based learning is extremely flexible, and has been used (among other tasks) for part-of-speech tagging [Brill, 1994], prepositional phrase attachment [Brill and Resnik, 1994], and parsing [Brill, 1993b].

Maximum entropy models are equally flexible in the kinds of evidence they allow and the types tasks they can perform. In the maximum entropy framework, we do not specify the space of possible transformations, but instead specify, in a very similar manner, the space of possible features. Maximum entropy models differ in that for each context b, they return a probability distribution over the possible outcomes, whereas a transformation based learner returns only an outcome.

3.4 Decision Lists

[Yarowsky, 1996] applies the learning technique of decision lists to the natural language problem of word sense disambiguation, using supervised and unsupervised techniques. The decision lists in [Yarowsky, 1996] effectively rank different pieces of evidence by reliability, so that unknown test events are classified by the single most reliable piece of evidence available. In our notation, if our space of outcomes consists of two elements, i.e., $\mathcal{A} = \{a', a''\}$, the reliability of each contextual predicate cp_i is given by the absolute value of the conditional log-likelihood ratio:

$$| \log \frac{p(a'|cp_i(b) = true)}{p(a''|cp_i(b) = true)} |$$

This ratio is used to create a sorted list of contextual predicates and outcomes $\{(cp_1, a_1) \ldots (cp_n, a_n)\}$, such that cp_1 has the highest log-likelihood ratio, and where a_i is the most probable outcome given cp_i, i.e. $a_i = \arg\max_{a \in \{a', a''\}} p(a|cp_i(b) = true)$. When classifying a test case b, the decision list technique chooses the outcome a_i that corresponds to the first predicate cp_i in the list such that $cp_i(b) = true$. The conditional probabilities used in the ratio must be smoothed when this technique is applied to word sense disambiguation, see [Yarowsky, 1996] for details. The decision list technique allows the experimenter to use many diverse forms of contextual evidence, but in the end chooses the outcome based on a *single* piece of reliable evidence.

Maximum entropy models can use equally diverse forms of evidence, but differ greatly in that their probability estimates depend on *many* pieces of evidence, and not just the single best one. [Yarowsky, 1996] argues that using the single best piece of evidence suffices to achieve high accuracies for word sense disambiguation, but also notes that further research is needed to validate this claim for other tasks.

3.5 Interpolation

Linear interpolation is a popular way to combine the estimates of derived from various pieces of evidence. For example, it has been used extensively in language modeling, in which the goal is to compute $P(w_i|w_{i-1}w_{i-2})$ by combining the estimates of several component distributions:

$$P(w_i|w_{i-1}w_{i-2}) = \lambda_1 p_1(w_i) + \lambda_2 p_2(w_i|w_{i-1}) + \lambda_3 p_3(w_i|w_{i-1}w_{i-2})$$

where $\lambda_i \geq 0$, and $\sum_{i=1}^{3} \lambda_i = 1$ Each component distribution p_i is estimated straight from the raw counts of the training data, and each λ_i is effectively a "weight" that reflects the importance of its corresponding component probability distribution. The weights are computed to maximize the likelihood of held-out data, see [Jelinek, 1990] for details. The technique can be generalized to combine any number of probability models:

$$p(a|b) = \sum_i \lambda_i p_i(a|cp_i(b) = true)$$

Here, $p_i(a|cp_i(b) = true)$ is the conditional probability distribution derived from the counts of $(cp_i(b), a)$ in the training set, and each predicate cp_i is associated with a λ_i, that weights the estimate $p_i(a|cp_i(b) = true)$ for $a \in \mathcal{A}$. The interpolation technique makes no assumptions on the underlying nature of the models that it is combining, and it is therefore a very general method for integrating evidence.

Maximum entropy models have the same level of generality as interpolation techniques, but differ in that any weight α_j and feature f_j are associated with *both* a contextual predicate cp and an outcome a. The weights in the maximum entropy model are somewhat "finer-grained" than in the interpolation model, which associates weights with only the predicates, and not the outcomes.

3.6 Decomposable Models

Decomposable models have been used for word sense disambiguation in [Bruce and Weibe, 1994, Pedersen et al., 1997] and also for prepositional phrase attachment in [Kayaalp et al., 1997]. Such models can be expressed as a product of the marginal probabilities of the interdependent variables, scaled by the marginal probabilities of the variables that are common to two or more terms. In our notation, if we are given three contextual predicates cp_1, cp_2, cp_3 such cp_1 and cp_2 are interdependent, and cp_3 is conditionally independent from cp_1 and cp_2, then the probability $p(a, cp_1, cp_2, cp_3,)$ is written as:

$$p(a, cp_1, cp_2, cp_3) = \frac{p_1(a, cp_1, cp_2) p_2(a, cp_3)}{p_3(a)}$$

(Here the event $cp_i(b) = true$ is abbreviated as simply cp_i.) No iterative parameter estimation algorithm is necessary to implement this algorithm; the relevant marginal probabilities p_1, p_2, p_3 can be obtained directly from the counts in the training data. In order to compute the joint probability given by a decomposable model, the interdepencies of the contextual predicates must either be known *a priori*, or must be induced automatically, as in [Pedersen et al., 1997]. Furthermore, the contextual predicates may be interdependent in such a way that prohibits further decomposition of the joint probability.

Maximum entropy models differ from decomposable models in how they handle interdependence among the features. In the maximum entropy framework, interdependencies are expressed through features, and not through the form of the model. However, in order to account for interdependencies expressed in the features, maximum entropy models require a computationally expensive iterative parameter estimation algorithm. Furthermore, the maximum entropy framework is more general, in that it can handle interdependencies that may not be expressible as decomposable models.

3.7 Logistic Regression Models

Logistic regression, as described in [Hosmer and Lemeshow, 1989], is a common technique for modeling the effects of one or more explanatory variables on some binary-valued outcome variable. For example, *success* and *failure* are commonly used outcomes, and the probability that the observations $\vec{x} = \{x_1 \ldots x_j\}$ of the explanatory variables indicate success is given by $q(\vec{x})$,

$$q(\vec{x}) = \frac{e^{g(\vec{x})}}{1 + e^{g(\vec{x})}} \tag{3.2}$$

where

$$g(\vec{x}) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$$

and where the x_j's are real-valued observations, and the β_j's are real-valued parameters. Likewise, the probability of the failure outcome is $1 - q(\vec{x})$. The logistic regression model form above is a special case of the maximum entropy model form (2.1), and we show below how to implement a logistic regression model under the maximum entropy framework. We assume a space of two outcomes $\mathcal{A} = \{0, 1\}$ that represent failure and success, respectively, and we further assume that the features are real-valued, and not binary-valued. We use a feature f_0

$$f_0(a,b) = \begin{cases} 1 & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

and the features $f_1 \ldots f_k$ of the format

$$f_j(a,b) = \begin{cases} x_j & \text{if } a = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where x_j is an observation of some explanatory variable (which presumably exists in the context b). The probability that the observations lead to success, or $p(a = 1|b)$, is given by:

$$\begin{aligned} p(a = 1|b) &= \frac{\prod_{j=0}^{k} \alpha_j^{f_j(1,b)}}{\prod_{j=0}^{k} \alpha_j^{f_j(0,b)} + \prod_{j=0}^{k} \alpha_j^{f_j(1,b)}} \\ &= \frac{e^{g(b)}}{1 + e^{g(b)}} \end{aligned}$$

where

$$g(b) = \beta_0 f_0(1,b) + \sum_{j=1}^{k} \beta_j f_j(1,b)$$

and where $\beta_j = \ln \alpha_j$, $f_0(1,b) = 1$, and $f_j(1,b)$ is defined to be x_j, for $j > 0$. Thus, if the features are defined as in (3.3) and (3.4), a maximum entropy model probability model obtained with the GIS algorithm is equivalent to a logistic regression model (3.2) in which the parameters are obtained with maximum likelihood estimation. However, while the above simulation of logistic regression assumes real-valued features and binary-valued outcomes, the implementation in this thesis differs in that it assumes binary-valued features and multiple-valued outcomes.

3.8 Conclusion

We use the maximum entropy framework for the tasks in this thesis because it offers some important advantages over other techniques. It allows more flexible features than the naive Bayes and decomposable probability models (at the expense of parameter estimation), and is capable of using more evidence for each prediction than the decision list technique. It is in theory equally flexible as linear interpolation, but studies in language modeling have shown that maximum entropy techniques perform better in practice, e.g., see [Rosenfeld, 1996]. Transformation-based learning is also a flexible and equally powerful technique when the goal is to find a single classification without a probability, but its non-probabilistic nature makes it difficult to rank *sequences* of classifications, as we need to do in Chapters 5 and 6. Logistic regression models are designed for problems with binary-valued outcomes, and are not suited for natural language tasks like tagging and parsing that require probability models with multiple-valued outcomes. Decision tree probability models have successfully been scaled up to attack the natural language parsing problem in previous work, such as [Black et al., 1993, Jelinek et al., 1994, Magerman, 1995], but have relied heavily on the word clustering technique of [Brown et al., 1992]. We believe that this clustering technique—based on contiguous word bigrams, and designed for n-gram language modeling— does not preserve the information necessary for highly accurate syntactic and semantic disambiguation. We therefore use the maximum entropy framework because it allows us to use the words directly, without the concern of data fragmentation. Furthermore, using a direct representation of words eliminates any harmful assumptions imposed by the clustering, and gives us the option of using the same kind of information used in the vast number of traditional approaches to natural language processing. Our hypothesis is that a direct representation of words under the maximum entropy framework will yield more accurate results than using a clustered representation with decision trees. Furthermore, we believe that the maximum entropy technique is a theoretically more compelling way to combine evidence than the other techniques reviewed here, and we wish to test if the theory will manifest itself in practice with better prediction accuracy.

Chapter 4

Sentence Boundary Detection

(This chapter represents joint work with Jeffrey C. Reynar of the University of Pennsylvania.)

4.1 Introduction

The task of identifying sentence boundaries in raw text has only recently received serious attention in the computational linguistics literature. Most natural language tools like part-of-speech taggers and parsers, including the ones discussed in this thesis, assume that the text has already been divided into sentences, and do not discuss algorithms for dividing it accurately.

On first glance, it may appear that postulating a sentence boundary for every occurrence of a potential sentence-final punctuation mark, such as *.*, *?*, and *!*, is sufficient to accurately divide text into sentences. However, these punctuation marks are not used exclusively to mark sentence breaks. For example, embedded quotations may contain any of the sentence-ending punctuation marks and *.* is used as a decimal point, in e-mail addresses, to indicate ellipsis and in abbreviations. Both *!* and *?* are somewhat less ambiguous but appear in proper names and may be used multiple times for emphasis to mark a single sentence boundary.

Lexically-based rules could be written and exception lists used to disambiguate the difficult cases described above. However, the lists will never be exhaustive, and multiple rules may interact badly since punctuation marks exhibit absorption properties. Sites which logically should be marked with multiple punctuation marks will often only have one ([Nunberg, 1990] as summarized in [White, 1995]). For example, a sentence-ending abbreviation will most likely not be followed by an additional period if the abbreviation already contains one (e.g. note that *D.C* is followed by only a single . in *The president lives in Washington, D.C.*).

The manual approach of writing rules appears to be both difficult and time-consuming, due to the large number of lexically-based rules that would need to be written, and due to the rule interactions that would need to be resolved. As an alternative, this chapter presents a solution based on a maximum entropy model which requires a few hints about what information to use and a corpus annotated with sentence boundaries. The model trains easily and performs comparably to systems that require vastly more information.

4.2 Previous Work

The most recent work on sentence boundary detection is [Palmer and Hearst, 1997], which describes a system architecture called SATZ and also includes a thorough review of other work related to sentence boundary detection. The SATZ architecture uses either a decision tree or a neural network to disambiguate sentence boundaries. The neural network achieves 98.5% accuracy on a corpus of *Wall Street Journal* articles using a lexicon which includes part-of-speech (POS) tag information. By increasing the quantity of training data and decreasing the size of their test corpus, [Palmer and Hearst, 1997] reports an accuracy of 98.9% with the neural network, and 99% with the decision tree. All the results presented in this chapter use their their initial, larger test corpus.

[Riley, 1989] describes a decision-tree based approach to the problem. Performance of this approach on the Brown corpus is 99.8%, using a model learned from a corpus of 25 million words. [Liberman and Church, 1992] suggest that a system could be quickly built to divide newswire text into sentences with a nearly negligible error rate, but do not actually build such a system.

4.3 Maximum Entropy Models for Sentence Boundary Identification

This chapter present two systems for identifying sentence boundaries, both based on maximum entropy models. One is targeted at high performance and uses some knowledge about the structure of English financial newspaper text which may not be applicable to text from other genres or in other languages. The other system uses no domain-specific knowledge and is aimed at being portable across English text genres and Roman alphabet languages.

Potential sentence boundaries are identified by scanning the text for sequences of characters separated by whitespace (tokens) containing one of the symbols *!*, *.* or *?*. The systems use information about the token containing the potential sentence boundary, as well as contextual information about the tokens immediately to the left and to the right. Wider contexts did not improve performance and were therefore omitted.

4.3.1 Outcomes

The outcomes of the probability model are `yes` and `no`, where `yes` denotes that a potential sentence boundary is an actual sentence boundary, and `no` denotes that it isn't a an actual sentence boundary.

4.3.2 Contextual Predicates

We call the token containing the symbol which marks a putative sentence boundary the Candidate. The portion of the Candidate preceding the potential sentence boundary is called the Prefix and the portion following it is called the Suffix. The system that focused on maximizing performance used the following hints, or contextual "templates":

- The Prefix
- The Suffix
- The presence of particular characters in the Prefix or Suffix
- Whether the Candidate is an honorific (e.g. *Ms.*, *Dr.*, *Gen.*)

- Whether the Candidate is a corporate designator (e.g. *Corp., S.p.A., L.L.C.*)
- Features of the word left of the Candidate
- Features of the word right of the Candidate

The templates specify only the form of the information. The *exact* set of contextual predicates used by the maximum entropy model for the potential sentence boundary marked by . in *Corp.* in Example 1 below would be: PreviousWordIsCapitalized, Prefix=*Corp*, Suffix=NULL, PrefixFeature=CorporateDesignator.

(1) ANLP Corp. chairman Dr. Smith resigned.

The highly portable system uses only the identity of the Candidate and its neighboring words, and a list of abbreviations induced from the training data.[1] Specifically, the "templates" used are:

- The Prefix
- The Suffix
- Whether the Prefix or Suffix is on the list of induced abbreviations
- The word left of the Candidate
- The word right of the Candidate
- Whether the word to the left or right of the Candidate is on the list of induced abbreviations

The information this model would use for Example 1 would be: PreviousWord=*ANLP*, FollowingWord=*chairman*, Prefix=*Corp*, Suffix=NULL, PrefixFeature=InducedAbbreviation.

The abbreviation list is automatically produced from the training data, and the contextual questions are also automatically generated by scanning the training data with question templates. As a result, no hand-crafted rules or lists are required by the highly portable system and it can be easily re-trained for other languages or text genres.

[1] A token in the training data is considered an abbreviation if it is preceded and followed by whitespace, and it contains a . that is *not* a sentence boundary.

4.3.3 Feature Selection and Decision Rule

For each potential sentence boundary token (., *?*, and *!*), we wish to estimate a joint probability distribution p of it and its surrounding context occurring as an actual sentence boundary. The probability distribution used here is a maximum entropy model identical to equation 2.1:

$$p(a|b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}$$

The contextual predicates deemed useful for sentence-boundary detection, which we described earlier, are encoded in the model using features. For example, a useful feature might be:

$$f_j(a,b) = \begin{cases} 1 & \text{if Prefix}(b) = \mathit{Mr}\ \&\ a = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

This feature will allow the model to discover that the period at the end of the word *Mr.* seldom occurs as a sentence boundary. Therefore the parameter corresponding to this feature will hopefully boost the probability $p(\texttt{no}|b)$ if the Prefix is *Mr.* All features occurring 10 times or more in the training data are retained in the model, and the model parameters are estimated with the *Generalized Iterative Scaling* [Darroch and Ratcliff, 1972] algorithm, described in Section 2.6.

All experiments use a simple decision rule to classify each potential sentence boundary: a potential sentence boundary in the context b is an actual sentence boundary if and only if $p(\texttt{yes}|b) > .5.$,

4.4 System Performance

	WSJ	Brown
Sentences	20478	51672
Candidate P. Marks	32173	61282
Accuracy	98.8%	97.9%
False Positives	201	750
False Negatives	171	506

Table 4.1: Our best performance on two corpora.

The system was trained on 39441 sentences (898737 words) of *Wall Street Journal* text from sections 00 through 24 of the second release of the Penn Treebank[2] [Marcus et al., 1994]. We corrected punctuation mistakes and erroneous sentence boundaries in the training data. Performance figures for our best performing system, which used a hand-crafted list of honorifics and corporate designators, are shown in Table 4.1. The first test set, WSJ, is Palmer and Hearst's initial test data and the second is the entire Brown corpus. We present the Brown corpus performance to show the importance of training on the genre of text on which testing will be performed. Table 4.1 also shows the number of sentences in each corpus, the number of candidate punctuation marks, the accuracy over potential sentence boundaries, the number of false positives and the number of false negatives. Performance on the WSJ corpus was, as we expected, higher than performance on the Brown corpus since we trained the model on financial newspaper text.

Possibly more significant than the system's performance is its portability to new domains and languages. The trimmed down system which only uses information derived from the training corpus performs nearly as well on the same test sets as the previous system, as shown in Table 4.2.

Test Corpus	Accuracy	False Positives	False Negatives
WSJ	98.0%	396	245
Brown	97.5%	1260	265

Table 4.2: Performance on the same two corpora using the highly portable system.

Since 39441 training sentences is considerably more than might exist in a new domain or a language other than English, we experimented with the quantity of training data

[2]We did not train on files which overlapped with Palmer and Hearst's test data, namely sections 03, 04, 05 and 06.

	Number of sentences in training corpus						
	500	1000	2000	4000	8000	16000	39441
Best performing	97.6%	98.4%	98.0%	98.4%	98.3%	98.3%	98.8%
Highly portable	96.5%	97.3%	97.3%	97.6%	97.6%	97.8%	98.0%

Table 4.3: Performance on *Wall Street Journal* test data as a function of training set size for both systems.

required to maintain performance. Table 4.3 shows performance on the WSJ corpus as a function of training set size using the best performing system and the more portable system. As can seen from the table, performance degrades as the quantity of training data decreases, but even with only 500 example sentences performance is better than the baselines of 64.0% if a sentence boundary is guessed at every potential site and 78.4% if only token-final instances of sentence-ending punctuation are assumed to be boundaries.

4.5 Conclusions

This chapter has described an approach to identifying sentence boundaries which performs comparably to other state-of-the-art systems that require vastly more resources. For example, the system of [Riley, 1989] performs better, but trains from the Brown corpus and uses thirty times as much data as our system. Also, the system of [Palmer and Hearst, 1997] requires POS tag information, which limits its use to those genres or languages for which there are either POS tag lexica or POS tag annotated corpora that could be used to train automatic taggers. In comparison, system in this chapter does not require POS tags or any supporting resources beyond the sentence-boundary annotated corpus. It is therefore easy and inexpensive to retrain this system for different genres of text in English and text in other Roman-alphabet languages. Furthermore, we showed that a small training corpus is sufficient for good performance, and we estimate that annotating enough data to achieve good performance would require only several hours of work, in comparison to the many hours required to generate POS tag and lexical probabilities.

4.6 Acknowledgments

This chapter is based on [Reynar and Ratnaparkhi, 1997], and represents joint work with Jeffrey C. Reynar of the University of Pennsylvania.

Both Jeff and I thank David Palmer and Marti Hearst for giving us the data used in their sentence detection experiments.

Chapter 5

Part-of-Speech Tag Assignment

5.1 Introduction

Many natural language tasks require the accurate assignment of Part-Of-Speech (POS) tags to previously unseen text. Due to the availability of large corpora which have been manually annotated with POS information, many taggers use annotated text to "learn" either probability distributions or rules and use them to automatically assign POS tags to unseen text.

This chapter presents a POS tagger implemented under the maximum entropy framework that learns a probability distribution for tagging from manually annotated data, namely, the Wall Street Journal corpus of the Penn Treebank project[Marcus et al., 1994], Since most realistic natural language applications must process words that were never seen before in training data, all experiments in this chapter are conducted on test data that include unknown words.

Several recent papers[Brill, 1994, Magerman, 1995] have reported 96.5% tagging accuracy on the Wall St. Journal corpus. The experiments in this chapter test the hypothesis that *better use of context will improve the accuracy.* A maximum entropy model is well-suited for such experiments since it combines diverse forms of contextual information in a principled manner. This chapter discusses the features used for POS tagging and the experiments on the Penn Treebank Wall St. Journal corpus. It then discusses the consistency problems discovered during an attempt to use specialized features on the word context. Lastly, the results in this chapter are compared to those from previous work on POS tagging.

5.2 The Probability Model

The probability $p(a|b)$ represents the conditional probability of a tag $a \in \mathcal{A}$, given some context or history $b \in B$, where $\mathcal{A}$ is the set of allowable tags, and where $\mathcal{B}$ is the set of possible word and tag contexts. The probability model is identical to equation 2.1:

$$p(a|b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}$$

where as usual, each parameter α_j corresponds to a feature f_j. Given a sequence of words $\{w_1, \ldots, w_n\}$ and tags $\{a_1, \ldots a_n\}$ as training data, we define b_i as the context available when predicting a_i.

5.3 Features for POS Tagging

The conditional probability of a history b and tag a is determined by those parameters whose corresponding features are active, i.e., those α_j such that $f_j(a,b) = 1$. A feature, given (a,b), may activate on any word or tag in the history b, and must encode any information that might help predict a, such as the spelling of the current word, or the identity of the previous two tags. For example, define the contextual predicate `currentsuffix_is_ing`(b) to return *true* if the current word in b ends with the suffix "ing". A useful feature might be

$$f_j(a, b_i) = \begin{cases} 1 & \text{if } \texttt{currentsuffix\_is\_ing}(b_i) = \textit{true} \ \& \ a = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

If the above feature exists in the feature set of the model, its corresponding model parameter will contribute towards the probability $p(a|b_i)$ when w_i ends with "`ing`" and when $a =$`VBG`[1]. Thus a model parameter α_j effectively serves as a "weight" for a certain contextual predictor, in this case the suffix "`ing`", towards the probability of observing a certain tag, in this case a `VBG`.

[1] `VBG` is the Penn treebank POS tag for progressive verb.

Condition	Contextual Predicates
w_i is not rare	$w_i = X$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$
	X is suffix of w_i, $\|X\| \leq 4$
	w_i contains number
	w_i contains uppercase character
	w_i contains hyphen
$\forall\ w_i$	$t_{i-1} = X$
	$t_{i-2}t_{i-1} = XY$
	$w_{i-1} = X$
	$w_{i-2} = X$
	$w_{i+1} = X$
	$w_{i+2} = X$

Table 5.1: Contextual Predicates on the context b_i

Word:	the	stories	about	well-heeled	communities	and	developers
Tag:	DT	NNS	IN	JJ	NNS	CC	NNS
Position:	1	2	3	4	5	6	7

Table 5.2: Sample Data

5.3.1 Contextual Predicates

The contextual predicates are generated automatically from the training data scanning each b_i with the "templates" in Table 5.1.

The generation of contextual predicates for tagging unknown words relies on the hypothesized distinction that "rare" words[2] in the training set are similar to unknown words in test data, with respect to how their spellings help predict their tags. Our technique of using rare words in training data for tagging unknown words in test data was developed independently from [Baayen and Sproat, 1996], who also observe that POS tags of words that occur once (the *hapax legomena*) are reliable predictors for POS tags of unknown words. The rare word predicates in Table 5.1, which look at the word spellings, will apply to both rare words and unknown words in test data.

[2]A "rare" word here denotes a word which occurs less than 5 times in the training set. The count of 5 was chosen by subjective inspection of words in the training data.

$w_i =$ `about`
$w_{i-1} =$ `stories`
$w_{i-2} =$ `the`
$w_{i+1} =$ `well-heeled`
$w_{i+2} =$ `communities`
$t_{i-1} =$ `NNS`
$t_{i-2}t_{i-1} =$ `DT NNS`

Table 5.3: Contextual Predicates Generated From b_3 (for tagging `about`) from Table 5.2

$w_{i-1} =$ `about`
$w_{i-2} =$ `stories`
$w_{i+1} =$ `communities`
$w_{i+2} =$ `and`
$t_{i-1} =$ `IN`
$t_{i-2}t_{i-1} =$ `NNS IN`
prefix(w_i)=`w`
prefix(w_i)=`we`
prefix(w_i)=`wel`
prefix(w_i)=`well`
suffix(w_i)=`d`
suffix(w_i)=`ed`
suffix(w_i)=`led`
suffix(w_i)=`eled`
w_i contains hyphen

Table 5.4: Contextual Predicates Generated From b_4 (for tagging `well-heeled`) from Table 5.2

For example, Table 5.2 contains an excerpt from training data while Table 5.3 contains the contextual predicates generated while scanning b_3, in which the current word is `about`. Table 5.4 contains predicates generated while scanning (b_4), in which the current word, `well-heeled`, occurs 3 times in training data and is therefore classified as "rare".

5.3.2 Feature Selection

The behavior of a feature that occurs very sparsely in the training set is often difficult to predict, since its statistics may not be reliable. Therefore, the model uses the heuristic that any feature which occurs less than 10 times in the data is unreliable, and ignores features whose counts are less than 10. Specifically, any contextual predicate cp that returned true in the presence of a particular prediction a' more than 10 times[3] is used as a feature f in the model, with the form:

$$f(a,b) = \begin{cases} 1 & \text{if } cp(b) = true \text{ and } a = a' \\ 0 & \text{otherwise} \end{cases}$$

5.4 Testing the Model

The test corpus is tagged one sentence at a time. The testing procedure requires a search to enumerate the candidate tag sequences for the sentence, and the tag sequence with the highest probability is chosen as the answer.

5.4.1 Search Algorithm

The search algorithm is a top K breadth first search (BFS); it is similar to a "beam search" and maintains, as it sees a new word, the K highest probability tag sequence candidates up to that point in the sentence. Given a sentence $\{w_1 \ldots w_n\}$, a tag sequence candidate $\{a_1 \ldots a_n\}$ has conditional probability:

$$P(a_1 \ldots a_n | w_1 \ldots w_n) = \prod_{i=1}^{n} p(a_i|b_i)$$

where b_i is the history corresponding to the ith word.

[3]Except for features that look only at the current word, i.e., features of the form $w_i =$`<word>` and $t_i =$`<TAG>`. A cutoff of 5, corresponding to the definition of "rare" words, was used for this kind of feature.

In addition the search procedure consults a *tag dictionary*, which is automatically constructed from the training data and whose entries have the form:

$$word\ t_1 \ldots t_n$$

where *word* is a word from the training set, and $t_1 \ldots t_n$ are the tags that *word* occurs with in the training set. When the search procedure needs to tag a word w, and w exists in the tag dictionary, only the tags from w's entry in the tag dictionary are considered as tag candidates for w. If w is not in the tag dictionary, the search procedure explores all tags in the tagset when tagging w. Table 5.5 describes the search procedure in more detail. The running time is dominated by the inner loop containing the `insert` function, which must, for each word, insert at most KT sequences into the heap where each insertion costs at most $O(\log KT)$, where T is the size of the tagset, and K is the number of tag sequences to maintain. Hence the running time on an N word sentence is $O(NKT \log KT)$.[4]

5.4.2 Experiments on the Wall St. Journal

In order to conduct tagging experiments, the Wall St. Journal data has been split into three contiguous sections, as shown in Table 5.6. The feature set and search algorithm were tested and debugged only on the training and development sets, and the official test result on the unseen test set is presented in Table 5.13. The performances of the tagging model with the "baseline" feature set (derived from Table 5.1), both with and without the Tag Dictionary, are shown in Table 5.7.

All experiments use $K = 5$; further increasing K does not significantly increase performance on the development set but adversely affects the speed of the tagger. Even though use of the tag dictionary gave an apparently insignificant (.12%) improvement in accuracy, it is used in further experiments since it reduces the average number of tags that are explored for each word, and thus significantly speeds up the tagger.

[4]Since the KT elements of a heap are known before the heap is created, a better implementation would have created the heap in one pass using the well-known linear time function heapify, in which case the search procedure's running time would have been $O(NKT)$. However, the current (asymptotically slower) implementation is more flexible in that it allows us to experiment with search strategies in which the heap elements are not all known when the heap is constructed.

```
advance:  s ⟶ s_1 ... s_m      /* Given tag sequence s, produce new
                               sequences s_1 ... s_m, each of length |s| + 1.
                               When necessary, this procedure consults
                               the tag dictionary.*/
insert:   s × h ⟶ void         /* inserts sequence s in heap h */
extract:  h ⟶ s                /* returns tag sequence in h with
                               highest score and also removes it from
                               h.*/

n = length of input sentence
N = 10
s = empty tag sequence
h_0 = empty heap        /* h_i contains tag sequences of length i */
insert(h_0, s)          /* initialize h_0 with empty sequence */

for i = 0 to n - 1
    sz = min(N, |h_i|)
    for j = 1 to sz
        s_1 ... s_m = advance( extract(h_i))
        for p = 1 to m
            insert(s_p, h_{i+1})

return extract(h_n)
```

Table 5.5: Tagger Search Procedure

DataSet	Sentences	Words	Unknown Words
Training	40000	962687	-
Development	8000	192826	6107
Test	5485	133805	3546

Table 5.6: WSJ Data Sizes

	Total Word Accuracy	Unknown Word Accuracy	Sentence Accuracy
Tag Dictionary	96.43%	86.23%	47.55%
No Tag Dictionary	96.31%	86.28%	47.38%

Table 5.7: Baseline Performance on Development Set

Word	Correct Tag	Proposed Tag	Frequency
about	RB	IN	393
that	DT	IN	389
more	RBR	JJR	221
up	IN	RB	187
that	WDT	IN	184
as	RB	IN	176
up	IN	RP	176
more	JJR	RBR	175
that	IN	WDT	159
about	IN	RB	144
that	IN	DT	127
out	RP	IN	126
that	DT	WDT	123
much	JJ	RB	118
yen	NN	NNS	117
chief	NN	JJ	116
up	RP	IN	114
ago	IN	RB	112
much	RB	JJ	111
out	IN	RP	109

Table 5.8: Top Tagging Mistakes on Training Set for Baseline Model

Number of "Difficult" Words	Development Set Performance
29	96.49%

Table 5.9: Performance of Baseline Model with Specialized Features

5.5 Specialized Features and Consistency

The maximum entropy model allows arbitrary binary-valued features on the context, so it can use additional specialized, i.e., word-specific, features to correctly tag the "residue" that the baseline features cannot model. Since such features typically occur infrequently, the training set consistency must be good enough to yield reliable statistics. Otherwise the specialized features will model noise and perform poorly on test data.

Such features can be designed for those words which are especially problematic for the model. The top errors of the model (over the training set) are shown in Table 5.8; clearly, the model has trouble with the words `that` and `about`, among others. As hypothesized in the introduction, better features on the context surrounding `that` and `about` should correct the tagging mistakes for these two words.

Specialized features for a given word are constructed by conjoining certain features in the baseline model with a question about the word itself. The features which ask about previous tags and surrounding words now additionally ask about the identity of the current word, e.g., a specialized feature for the word `about` in Table 5.3 could be:

$$f_j(a_i, b_i) = \begin{cases} 1 & \text{if } w_i = \texttt{about} \ \& \ t_{i-2}t_{i-1} = \texttt{DT NNS} \\ & \& \ a_i = \texttt{IN} \\ 0 & \text{otherwise} \end{cases}$$

where w_i is the current word in b_i, and where $t_{i-2}t_{i-1}$ are the previous two tags in b_i.

Table 5.9 shows the results of an experiment in which specialized features are constructed for "difficult" words, and are added to the baseline feature set. Here, "difficult" words are those that are mistagged a certain way at least 50 times when the training set is tagged with the baseline model. Using the set of 29 difficult words, the model performs at 96.49% accuracy on the Development Set, an insignificant improvement from the baseline accuracy of 96.43%. Table 5.10 shows the change in error rates on the Development

Word	# Baseline Model Errors	# Specialized Model Errors
that	246	207
up	186	169
about	110	120
out	104	97
more	88	89
down	81	84
off	73	78
as	50	38
much	47	40
chief	46	47
in	39	39
executive	37	33
most	23	34
ago	22	18
yen	18	17

Table 5.10: Errors on Development Set with Baseline and Specialized Models

Set for the frequently occurring "difficult" words. For most words, the specialized model yields little or no improvement, and for some, i.e., `more` and `about`, the specialized model performs worse.

The lack of improvement implies that either the feature set is still impoverished, or that the training data is inconsistent. A simple consistency test is to graph the POS tag assignments for a given word as a function of the article in which it occurs. Consistently tagged words should have roughly the same tag distribution as the article numbers vary. Figure 5.1 represents each POS tag with a unique integer and graphs the POS annotation of `about` in the training set as a function of the article# (the points are "scattered" to show density). As seen in figure 5.1, `about` is usually annotated with tag#1, which denotes `IN` (preposition), or tag#9, which denotes `RB` (adverb), and the observed probability of either choice depends heavily on the current article#. Upon further examination[5], the tagging distribution for `about` changes *precisely* when the annotator changes. Figure 5.2, which again uses integers to denote POS tags, shows the tag distribution of `about` as a function of annotator, and implies that the tagging errors for this word are due mostly

[5]The mapping from article to annotator is in the file `doc/wsj.wht` on the Treebank v.5 CDROM.

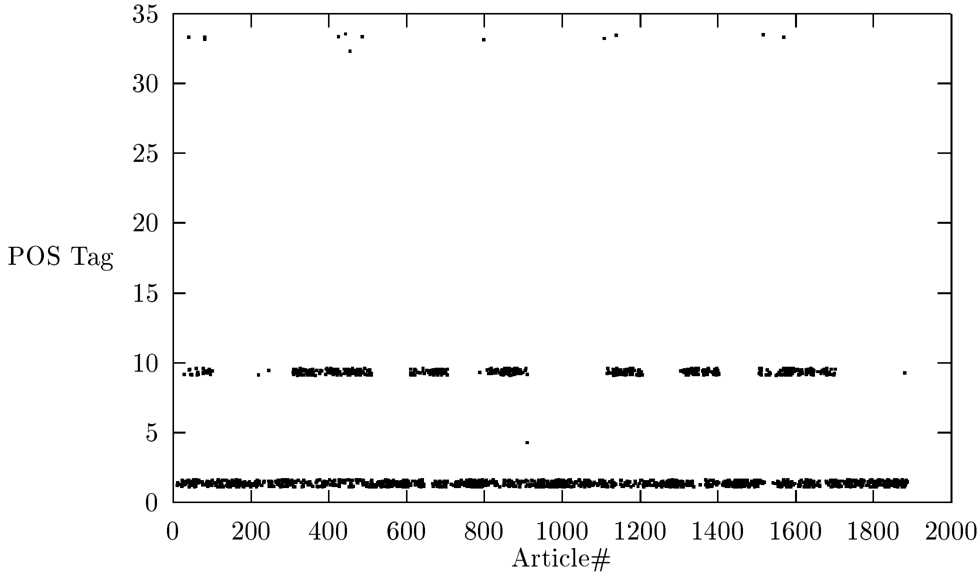


Figure 5.1: Distribution of Tags for the word “about” vs. Article#

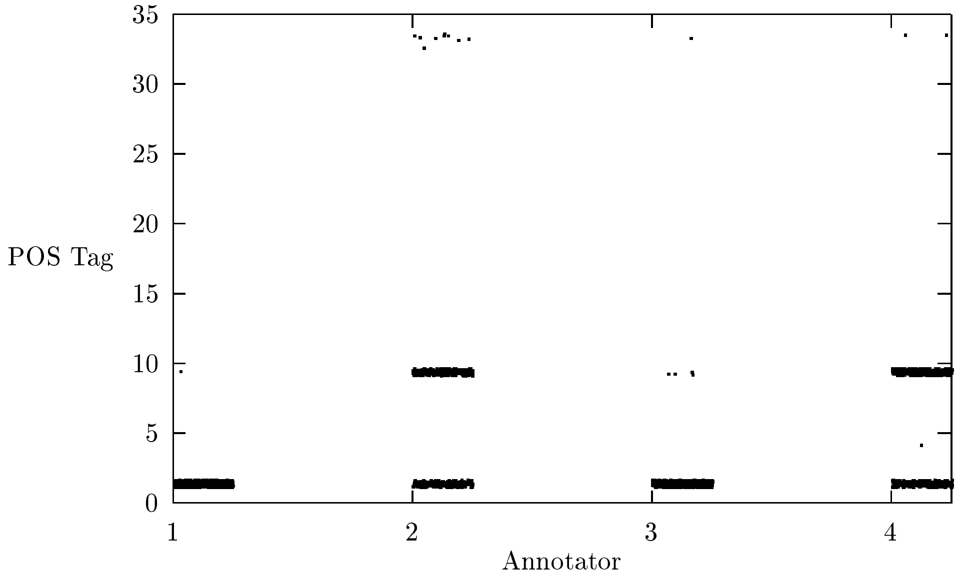


Figure 5.2: Distribution of Tags for the word “about” vs. Annotator

Training Size(words)	Test Size(words)	Baseline	Specialized
571190	44478	97.04%	97.13%

Table 5.11: Performance of Baseline & Specialized Model When Tested on Consistent Subset of Development Set

to inconsistent data. The words `ago`, `chief`, `down`, `executive`, `off`, `out`, `up` and `yen` also exhibit similar bias.

Thus specialized features may be less effective for those words affected by inter-annotator bias. A simple solution to eliminate inter-annotator inconsistency is to train and test the model on data that has been created by the *same* annotator. The results of such an experiment[6] are shown in Table 5.11. The total accuracy is higher, implying that the singly-annotated training and test sets are more consistent, and the improvement due to the specialized features is higher than before (.1%) but still modest, implying that either the features need further improvement or that *intra*-annotator inconsistencies exist in the corpus.

5.6 Experiments on other Corpora

The tagger has also been evaluated on the LOB corpus[Johansson, 1986], which contains samples of British English, and also on the CRATER corpus[Sánchez-León, 1994], which contains samples of Spanish in the telecommunications domain. The templates to create the baseline feature set (shown in Table 5.1) for the Wall St. Journal experiments were also used for both the LOB and CRATER corpus experiments. The performance of the maximum entropy tagger on these corpora with the baseline feature set is shown in Table 5.12.

The tagset of the CRATER corpus is very detailed and consists of over 500 tags; we mapped them down to a smaller set of 123 tags. The maximum entropy tagger was trained

[6] The single-annotator training data was obtained by extracting those articles tagged by "maryann" in the Treebank v.5 CDROM. This training data does not overlap with the Development and Test set used in the chapter. The single-annotator Development Set is the portion of the Development Set which has also been annotated by "maryann". The word vocabulary and tag dictionary are the same as in the baseline experiment.

Corpus	Accuracy		
	Word	Unknown Word	Sentence
LOB Corpus	97.43%	N/A	N/A
CRATER Corpus	97.7%	83.3%	60.4%

Table 5.12: Performance on the LOB corpus and CRATER corpus with baseline feature set

on 12k sentences from this corpus, and tested on the remaining 4k sentences; the results are shown in Table 5.12. The preprocessing, tokenization, and experiments on the LOB corpus are described elsewhere in [van Halteren et al., 1998].

5.7 Comparison With Previous Work

Most of the recent corpus-based POS taggers in the literature either use markov modeling techniques[Weischedel et al., 1993, Merialdo, 1994], statistical decision tree techniques[Jelinek et al., 1994, Magerman, 1995], or transformation based learning[Brill, 1994]. The maximum entropy tagger presented in this chapter combines the advantages of all these methods. It uses a rich feature representation, like transformation based learning, and generates a tag probability distribution for each tag decision, like decision tree and markov model techniques.

[Weischedel et al., 1993] provide the results from a battery of "tri-tag" markov model experiments, in which the probability $P(W,T)$ of observing a word sequence $W = \{w_1, w_2, \ldots, w_n\}$ together with a tag sequence $T = \{t_1, t_2, \ldots, t_n\}$ is given by:

$$\begin{aligned} P(T|W)P(W) &= p(t_1)p(t_2|t_1) \times \\ & \prod_{i=3}^{n} p(t_i|t_{i-1}t_{i-2}) \left(\prod_{1}^{n} p(w_i|t_i) \right) \end{aligned}$$

Furthermore, $p(w_i|t_i)$ for unknown words is computed by the following heuristic, which uses a set of 35 pre-determined endings:

$$\begin{aligned} p(w_i|t_i) &= p(\text{unknownword}|t_i) \times \\ & p(\text{capitalfeature}|t_i) \times \end{aligned}$$

$$p(\text{endings, hypenation}|t_i)$$

On the Wall St. Journal corpus, this approximation works as well as the maximum entropy model, giving 85% unknown word accuracy[Weischedel et al., 1993], despite its independence assumptions. However, as more diverse information sources are added, many of them are likely to be statistically dependent, and approximations that rely on independence assumptions may not adequately model the data. In contrast, the maximum entropy model combines diverse and non-local information sources without making any independence assumptions on the features.

A POS tagger is one component in the decision tree based statistical parsing system described in [Jelinek et al., 1994, Magerman, 1995]. The total word accuracy on Wall St. Journal data, 96.5%[Magerman, 1995], is similar to that presented in this chapter. However, these techniques require word classes[Brown et al., 1992] to help prevent data fragmentation, and a sophisticated smoothing algorithm to mitigate the effects of any fragmentation that occurs. Unlike decision trees, the maximum entropy training procedure does not recursively split the data, and hence does not suffer from unreliable counts due to data fragmentation. As a result, a word class hierarchy and smoothing algorithm are not required to achieve the same level of accuracy.

[Brill, 1994] presents transformation-based learning, a data-driven but non-probabilistic approach to POS tagging which also uses a rich feature representation, and performs at a total word accuracy of 96.5% and an unknown word accuracy of 85%[Brill, 1994]. The representation used in [Brill, 1994] is somewhat similar to the one used in this chapter. [Brill, 1994] looks at words ± 3 away from the current word, whereas the feature set in this chapter uses a window of ± 2. For unknown words, [Brill, 1994] uses a separate transformation-based learner and uses prefix/suffix additions and deletions, which are not used in this chapter. The tagger in this chapter, unlike [Brill, 1994], does not use a separate model for unknown words, and uses both features for tagging known words (such as the previous tag context) together with spelling features in order to tag unknown words. Transformation-based learning is non-probabilistic, it cannot be used as a probabilistic component in a larger model. In contrast, the tagger in this chapter provides a probability

Total Word Accuracy	Unknown Word Accuracy	Sentence Accuracy
96.63%	85.56%	47.51%

Table 5.13: Performance of Specialized Model on Unseen Test Data

for each tagging decision, which can be used in the probability calculation of any structure that is predicted over the POS tags, such as noun phrases, or entire parse trees, as will be demonstrated in Chapter 6.

While the claimed advantages of the maximum entropy tagger over other taggers are not realized on the Wall St. Corpus, they are apparently evident on the LOB corpus, since [van Halteren et al., 1998] reports that the maximum entropy tagger outperformed all the other taggers tested, including the transformation-based learning tagger and an HMM tagger. We suspect that all taggers have approached a performance limit (roughly 96.5%) on the Wall St. Journal due to the inherent noise in the corpus, and that the taggers have not yet approached a similar limit on the less noisy LOB corpus.

5.8 Conclusion

The implementation in this chapter is a state-of-the-art POS tagger, as evidenced by the 96.6% accuracy on unseen Wall St. Journal data, shown in Table 5.13. The model with specialized features does not perform much better than the baseline model, and further discovery or refinement of word-based features is difficult given the inconsistencies in the training data. A model trained and tested on data from a single annotator performs at .5% higher accuracy than the baseline model and should produce more consistent input for applications that require tagged text.

Chapter 6

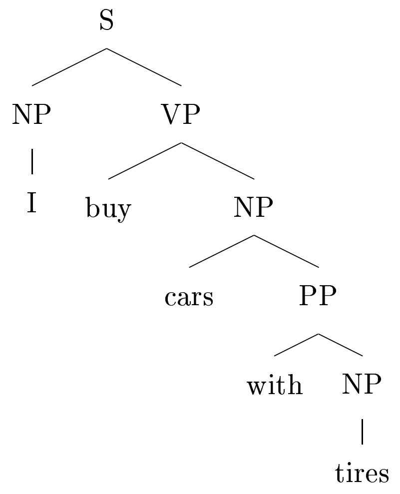
Parsing

6.1 Introduction

The task of a natural language parser is to take a sentence as input and return a syntactic representation that corresponds to the likely semantic interpretation of the sentence. For example, some parsers, given the sentence

I buy cars with tires

would return a parse tree in the format:

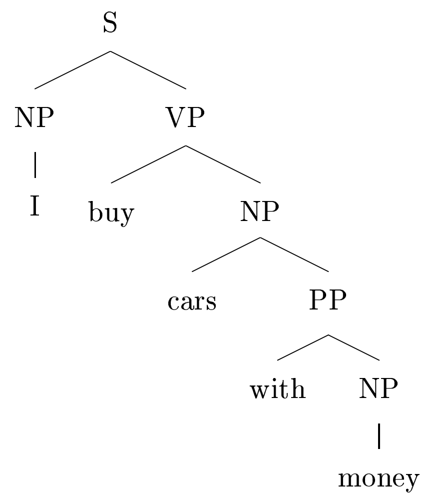


where the non-terminal labels denote the type of phrase (e.g., "PP" stands for prepositional phrase). Accurate parsing is difficult because subtle aspects of word meaning—from the parser's view—dramatically affect the interpretation of the sentence. For example, given the sentence

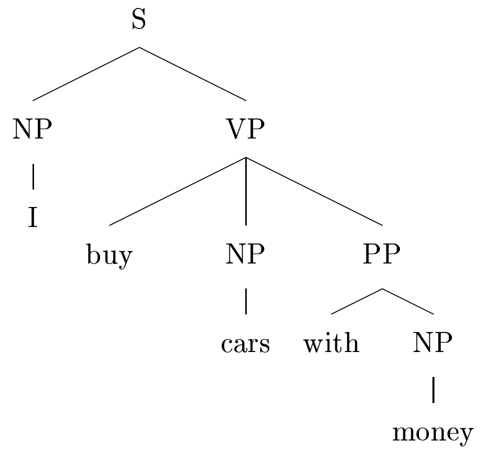
I buy cars with money

a parser might propose the following two parses

- (Unlikely:)



- (Likely:)



Both parses are grammatical, in the sense that a typical context free grammar for English will generate both structures, but only one corresponds to the likely interpretation of the sentence. A parser actually needs detailed semantic knowledge of certain key words in the sentence order to distinguish the correct parse; it needs to somehow know that *with money* refers to *buy* and not *car*.

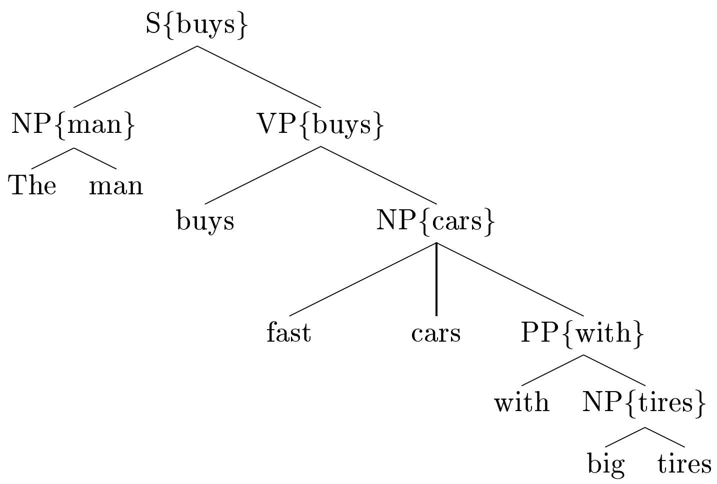


Figure 6.1: A parse tree annotated with head words

The parsers which currently show superior accuracies on freely occurring text are all classified as *statistical* or *corpus-based*, since they automatically learn syntactic and semantic knowledge for parsing from a large corpus of text, called a *treebank*, that has been manually annotated with syntactic information. In order to evaluate the accuracy of a statistical parser, we first train it on a subset of the treebank, test it on another non-overlapping subset, and then compare the labelled syntactic constituents it proposes with the labelled syntactic constituents in the annotation of the treebank. The labelled constituent accuracies of the best parsers approach roughly 90% when tested on freely occurring sentences in the Wall St. Journal domain.

6.2 Previous Work

Recent corpus-based parsers differ in the simplicity of their representation and the degree of supervision necessary, but agree in that they resolve parse structure ambiguities by looking at certain cooccurrences of constituent *head words* in the ambiguous parse. A head word of a constituent, informally, is the one word that best represents the meaning of the constituent, e.g., figure 6.1 shows a parse tree annotated with head words. Parsers vary greatly on how head word information is used to disambiguate possible parses for an input sentence. [Black et al., 1993] introduces *history-based* parsing, in which decision tree probability models, trained from a treebank, are used to score the different derivations of sentences produced by a hand-written grammar. [Jelinek et al., 1994, Magerman, 1995] also train history-based decision tree models from a treebank for use in a parser, but do not require an explicit hand-written grammar. The decision trees in [Black et al., 1993, Jelinek et al., 1994, Magerman, 1995] do not look at words directly, but instead represent words as bitstrings derived from an automatic clustering technique. In contrast, [Hermjakob and Mooney, 1997] uses a rich semantic representation when training decision tree and decision list techniques to drive parser actions.

Several other recent papers use statistics of pairs of head words in conjunction with chart parsing techniques to achieve high accuracy. The parsers in [Collins, 1996, Collins, 1997] use chart-parsing techniques and head word bigram statistics derived from a treebank. [Charniak, 1997] uses head word bigram statistics with a probabilistic context free grammar, while [Goodman, 1997] uses head word bigram statistics with a probabilistic feature grammar. [Collins, 1996, Goodman, 1997, Charniak, 1997, Collins, 1997] do not use general machine learning algorithms, but instead develop specialized statistical estimation techniques for their respective parsing tasks.

The parser in this paper attempts to combine the advantages of other approaches. It uses a natural and direct representation of words in conjunction with a general machine learning technique, maximum entropy modeling. We argue that the successful use of a simple representation with a general learning technique is the combination that both *minimizes human effort* and maintains state-of-the-art parsing accuracy.

6.3 Parsing with Maximum Entropy Models

The parser presented here constructs labelled syntactic parse trees with actions similar to those of a standard shift–reduce parser. The sequence of actions $\{a_1 \dots a_n\}$ that construct a completed parse tree T are called the *derivation* of T. There is no explicit grammar that dictates what actions are allowable; instead, all actions that lead to a well-formed parse tree are allowable and maximum entropy probability models are used to score each action. The maximum entropy models are trained by examining the derivations of the parse trees in a large, hand-corrected, corpus of example parse trees. The individual scores of the actions

Pass	Procedure	Actions	Description
First Pass	TAG	A POS tag in tag set	Assign POS Tag to word
Second Pass	CHUNK	`Start X`, `Join X`, `Other`	Assign Chunk tag to POS tag and word
Third Pass	BUILD	`Start X`, `Join X`, where `X` is a constituent label in label set	Assign current tree to start a new constituent, or to join the previous one
	CHECK	`Yes`, `No`	Decide if current constituent is complete

Table 6.1: Tree-Building Procedures of Parser

in a derivation are used to compute a score for the whole derivation, and hence the whole parse tree. When parsing a sentence, the parser uses a search procedure that efficiently explores the space of possible parse trees, and attempts to find the highest scoring parse tree.

6.3.1 Actions of the Parser

The actions of the parser are produced by *procedures*, that each take a derivation $d = \{a_1 \ldots a_n\}$, and predict some action a_{n+1} to create a new derivation $d' = \{a_1 \ldots a_{n+1}\}$. The actions of the procedures are designed so that any possible complete parse tree T has *exactly one* derivation.

The procedures are called TAG, CHUNK, BUILD, and CHECK, and are applied in three left-to-right passes over the input sentence; the first pass applies TAG, the second pass applies CHUNK, and the third pass applies BUILD and CHECK. The passes, the procedures they apply, and the actions of the procedures are summarized in table 6.1. Typically, the parser explores many different derivations when parsing a sentence, but for illustration purposes, figures 6.2–6.8 trace one possible derivation for the sentence "I saw the man with the telescope", using the constituent labels and part-of-speech (POS) tags of the University of Pennsylvania treebank[Marcus et al., 1994].

The actions of the procedures are scored with maximum entropy probability models that use information in the local context to compute their probability distributions. (A more detailed discussion of the probability models will occur in Section 6.3.2.) Using three passes instead of one pass allows the the use of more local context. For example, the model for the CHUNK procedure will have the output from TAG in its left and right context, and the models for the BUILD and CHECK procedures will have the output of TAG and CHUNK and their left and right contexts. If all these procedures were implemented in one left-to-right pass, the model for CHUNK would not have the output of TAG in its right context, and the models for BUILD and CHECK would not have the output of TAG and CHUNK in their right context.

First Pass

The first pass takes an input sentence, shown in figure 6.2, and uses TAG to assign each word a POS tag. The result of applying TAG to each word is shown in figure 6.3. The tagging phase is described in Chapter 5 in more detail. It is integrated into the parser's search procedure, so that the parser does not need to commit to a single POS tag sequence.

Second Pass

The second pass takes the output of the first pass and uses CHUNK to determine the "flat" phrase chunks of the sentence, where a phrase is "flat" if and only if it is a constituent whose children consist solely of POS tags. Starting from the left, CHUNK assigns each (word,POS tag) pair a "chunk" tag, either `Start X`, `Join X`, or `Other`. Figure 6.4 shows the result after the second pass. The chunk tags are then used for chunk detection, in which any consecutive sequence of words $w_m \ldots w_n$ $(m \leq n)$ are grouped into a "flat" chunk X if w_m has been assigned `Start X` and $w_{m+1} \ldots w_n$ have all been assigned `Join X`. The result of chunk detection, shown in figure 6.5, is a forest of trees and serves as the input to the third pass.

The granularity of the chunks, as well as the possible constituent labels of the chunks, are determined from the treebank that is used to train the parser. Examples of constituents that are marked as flat chunks in the Wall St. Journal domain of the Penn treebank include

Procedure	Actions	Similar Shift-Reduce Parser Action
CHECK	`No`	shift
CHECK	`Yes`	reduce α, where α is CFG rule of proposed constituent
BUILD	`Start X`, `Join X`	Determines α for subsequent reduce operations

Table 6.2: Comparison of BUILD and CHECK to operations of a shift-reduce parser

I saw the man with the telescope

Figure 6.2: Initial Sentence

noun phrases such as *a nonexecutive director*, adjective phrases such as *61 years old*, and quantifier phrases such as *about $ 370 million*.

The chunking in our second pass differs from other chunkers in the literature[Ramshaw and Marcus, 1995, Church, 1988] in that it finds chunks of all constituent labels, and not just noun phrase chunks. Our multi-pass approach is similar to the approach of the parser in [Abney, 1991], which also first finds chunks in one pass, and then attaches them together in the next pass.

PRP	VBD	DT	NN	IN	DT	NN
|	|	|	|	|	|	|
I	saw	the	man	with	the	telescope

Figure 6.3: The result after First Pass

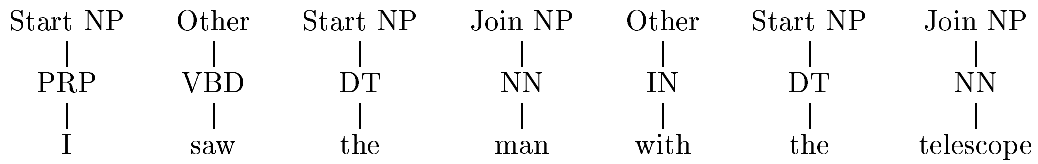


Figure 6.4: The result after Second Pass

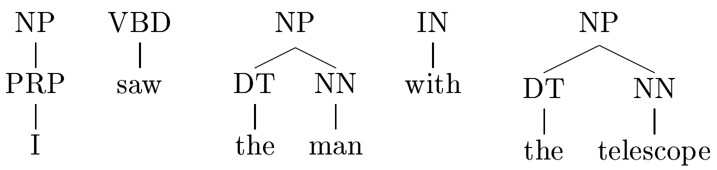


Figure 6.5: The result of chunk detection

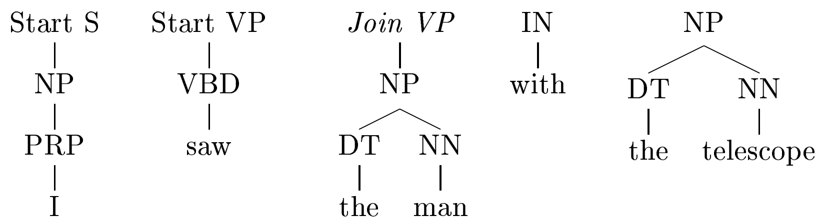


Figure 6.6: An application of BUILD in which `Join VP` is the action

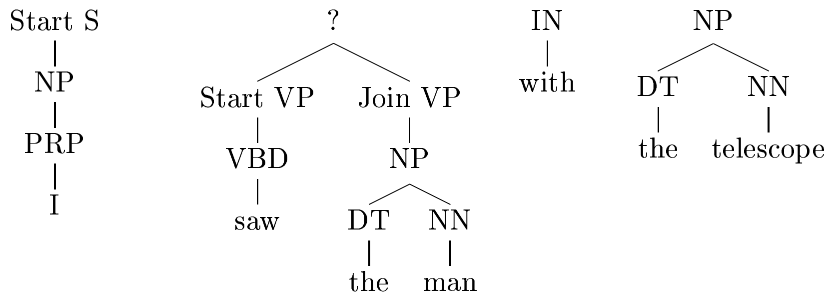


Figure 6.7: The most recently proposed constituent (shown under ?)

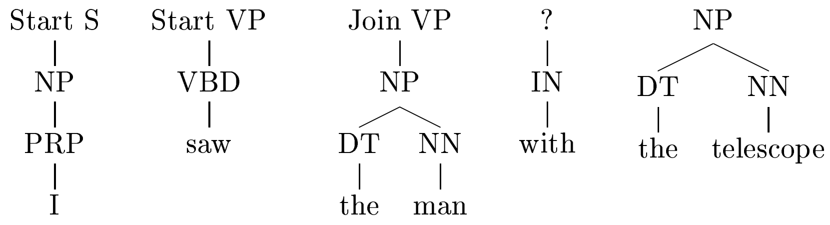


Figure 6.8: An application of CHECK in which `No` is the action, indicating that the proposed constituent in figure 6.7 is *not* complete. BUILD will now process the tree marked with ?

Third Pass

The third pass always alternates between the use of BUILD and CHECK, and completes any remaining constituent structure. BUILD decides whether a tree will start a new constituent or join the incomplete constituent immediately to its left. Accordingly, it annotates the tree with either `Start X`, where `X` is any constituent label, or with `Join X`, where `X` matches the label of the incomplete constituent immediately to the left. BUILD always processes the leftmost tree without any `Start X` or `Join X` annotation. Figure 6.6 shows an application of BUILD in which the action is `Join VP`. After BUILD, control passes to CHECK, which finds the most recently proposed constituent, and decides if it is complete. The most recently proposed constituent, shown in figure 6.7, is the rightmost sequence of trees $t_m \dots t_n$ ($m \leq n$) such that t_m is annotated with `Start X` and $t_{m+1} \dots t_n$ are annotated with `Join X`. If CHECK decides `yes`, then the proposed constituent takes its place in the forest as an actual constituent, on which BUILD does its work. Otherwise, the constituent is not finished and BUILD processes the next tree in the forest, t_{n+1}. CHECK always answers `no` if the proposed constituent is a "flat" chunk, since such constituents must be formed in the second pass. Figure 6.8 shows the result when CHECK looks at the proposed constituent in figure 6.7 and decides `No`. The third pass terminates when CHECK is presented a constituent that spans the entire sentence.

Table 6.2 compares the actions of BUILD and CHECK to the operations of a standard shift-reduce parser. The `No` and `Yes` actions of CHECK correspond to the shift and reduce actions, respectively. The important difference is that while a shift-reduce parser creates a constituent in one step (reduce α), the procedures BUILD and CHECK create it over several steps in smaller increments.

While the use of maximum entropy models together with shift–reduce parsing is novel (to our knowledge), shift–reduce parsing techniques have been popular in the natural language literature. [Aho et al., 1988] describe shift–reduce parsing techniques (for programming languages) in detail, [Marcus, 1980] uses shift–reduce parsing techniques for natural language, and [Briscoe and Carroll, 1993] describe probabilistic approaches to *LR parsing*, a type of shift–reduce parsing.

6.3.2 Maximum Entropy Probability Model

The parser uses a "history-based" approach [Black et al., 1993], in which a probability $p_X(a|b)$ is used to score an action a of procedure $X \in \{\text{TAG}, \text{CHUNK}, \text{BUILD}, \text{CHECK}\}$ depending on the context b that is available at the time of the decision. The conditional models p_X are estimated under the maximum entropy framework, as described in Chapter 2. The advantage of this framework is that we can use arbitrarily diverse information in the context b when computing the probability of an action a of some procedure X.

While any context b is a rich source of information, it is (in general) difficult to know *exactly* what information is useful for parsing. However, we would like to implement the following inexact intuitions about parsing:

- Using constituent head words is useful.
- Using combinations of head words is useful.
- Using less-specific information is useful.
- Allowing limited lookahead is useful.

The above intuitions are implemented in the maximum entropy framework as *features*, and each feature is assigned a "weight" which corresponds to how useful it is for modeling the data. We describe the outcomes, the contextual predicates, and the feature selection strategy for the parsing models p_X below, and furthermore show that a mere handful of guidelines are sufficient to completely describe the feature sets used by the parsing models. We also describe how the probability models p_X are used to compute the score of a parse tree.

Outcomes

The outcomes of the conditional probability models p_{TAG}, p_{CHUNK}, p_{BUILD} and p_{CHECK} are exactly the allowable actions of the TAG, CHUNK, BUILD, and CHECK procedures, listed in Table 6.1.

Contextual Predicates

The features in this chapter require *contextual predicates* to look at any information in the partial derivation or context. A contextual predicate has the form $cp : \mathcal{B} \rightarrow \{true, false\}$ and checks for the presence or absence of useful information in a context $b \in \mathcal{B}$ and returns true or false accordingly. In this implementation of the maximum entropy framework, every feature f has the format

$$f(a,b) = \begin{cases} 1 & \text{if } cp(b) = \texttt{true} \;\&\&\; a = a' \\ 0 & \text{otherwise} \end{cases}$$

and therefore must use a contextual predicate cp to express a cooccurrence relationship between some action a' and some linguistic fact about the context captured by cp. The contextual predicates for a procedure X are denoted by $\mathcal{CP}_X$, and Table 6.3 specifies the guidelines, or templates, for creating $\mathcal{CP}_X$, where $X \in \{$ TAG, CHUNK, BUILD, CHECK $\}$. The templates are only linguistic hints, in that they do not specify the information itself, but instead, specify *the location* of the useful information in a context b. The templates use indices relative to the tree that is currently being modified. For example, if the current tree is the 5th tree, $\mathsf{cons}(-2)$ looks at the constituent label, head word, and start/join annotation of the 3rd tree in the forest. The actual contextual predicates in $\mathcal{CP}_X$ are obtained automatically, by recording certain aspects of the context (specified by the templates) in which procedure X was used in the derivations of the trees in the treebank. For an example, an actual contextual predicate $cp \in \mathcal{CP}_{\text{BUILD}}$, derived (automatically) from the template $\mathsf{cons}(0)$, might be

$$cp(b) = \begin{cases} true & \text{if the 0th tree of } b \text{ has label "NP" and head word "he"} \\ false & \text{otherwise} \end{cases}$$

In order to obtain this predicate, there must exist a derivation in the manually parsed example sentences in which BUILD decides an action in the presence of some partial derivation b, such that the 0th tree of b had a constituent label "NP" and head word "he". Constituent head words are found, when necessary, with the algorithm in [Black et al., 1993, Magerman, 1995].

Contextual predicates which look at head words, or especially pairs of head words, may not be reliable predictors for the procedure actions due to their sparseness in the training set. Therefore, for each lexically based contextual predicate, there also exist one or more corresponding less specific contextual predicates which look at the same context, but *omit* one or more words. For example, the templates $\mathsf{cons}(0, 1^*)$, $\mathsf{cons}(0^*, 1)$, $\mathsf{cons}(0^*, 1^*)$ are the same as $\mathsf{cons}(0, 1)$ but omit references to the head word of the 1st tree, the 0th tree, and both the 0th and 1st tree, respectively. The less specific contextual predicates should allow the model to provide reliable probability estimates when the words in the history are rare. Less specific predicates are not enumerated in table 6.3, but their existence is indicated with a * and †. The *default* predicates in table 6.3 return true for any context and are the least specific (and most frequent) predicates; they should provide reasonable estimates when the model encounters a context in which every other contextual predicate is unreliable.

The contextual predicates attempt to capture the intuitions about parsing information discussed earlier. For example, predicates derived from templates like $\mathsf{cons}(0)$ look at constituent head words, while predicates derived from templates like $\mathsf{cons}(-1, 0)$ look at combinations of head words. Predicates derived from templates like $\mathsf{cons}(-1^*, 0)$ look at less specific information, while predicates derived from templates like $\mathsf{cons}(0, 1, 2)$ use limited lookahead. Furthermore, the information expressed in the predicates is always local to where the parsing action is taking place. The contextual predicates for TAG, discussed in Chapter 5, look at the previous 2 words and tags, the current word, and the following 2 words. The contextual predicates for CHUNK look at the previous 2 words, tags, and chunk labels, as well as the current and following 2 words and tags. BUILD uses head word information from the previous 2 and current trees, as well as the following 2 chunks, while CHECK looks at the surrounding 2 words and the head words of the children of the proposed constituent. The intuitions behind the contextual predicates are not linguistically deep, and as a result, the information necessary for parsing can be specified concisely with only a few templates.

Feature Selection

Feature selection refers to the process of choosing a useful subset of features $\mathcal{S}_X$ from the set of all possible features $\mathcal{P}_X$ for use in the maximum entropy model corresponding to procedure X. If $\mathcal{CP}_X$ are all the contextual predicates used to encode the training events $\mathcal{T}_X$, and $\mathcal{A}_X$ are the possible actions for procedure X, the set of possible features $\mathcal{P}_X$ for use in X's model are:

$$\begin{aligned}\mathcal{P}_X &= \{f \mid f(a,b) = \begin{cases} 1 & \text{if } cp(b) = \texttt{true} \text{ \&\& } a = a' \\ 0 & \text{otherwise} \end{cases} \\ &\quad \text{where } cp \in \mathcal{CP}_X \text{ and } a' \in \mathcal{A}_X\}\end{aligned}$$

Thus any contextual predicate cp that occurs with any action a' can potentially be a feature. However, many of these features occur infrequently, and are therefore not reliable sources of evidence since their behavior in the training events may not represent their behavior in unseen data. For example, it is unlikely that all of the contextual predicates in Table 6.9 would form reliable features.

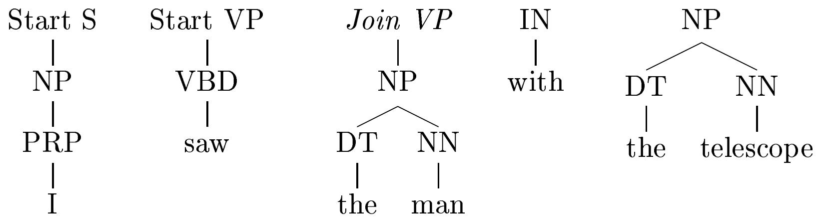
We use a very simple feature selection strategy: assume that any feature that occurs less than 5 times is noisy and discard it. Feature selection with a count cutoff does not yield a *minimal* feature set; many of the selected features will be redundant. However, in practice, it yields a feature set that is mostly noise-free with almost no computational expense. Therefore, the selected features for use in procedure X's model are

$$\begin{aligned}\mathcal{S}_X &= \{f \mid f(a,b) = \begin{cases} 1 & \text{if } cp(b) = \texttt{true} \text{ \&\& } a = a' \\ 0 & \text{otherwise} \end{cases} \\ &\quad \text{where } cp \in \mathcal{CP}_X \text{ and } a' \in \mathcal{A}_X,\ \sum_{(a,b)\in\mathcal{T}_X} f(a,b) \geq 5\}\end{aligned}$$

In this approach, the burden of deciding the contribution of each selected feature towards modeling the data falls to the parameter estimation algorithm.

Model	Categories	Description	Templates Used
TAG		See Chapter 5	
CHUNK	chunkandpostag$(n)^*$	The word, POS tag, and chunk tag of nth leaf. Chunk tag omitted if $n \geq 0$.	chunkandpostag(0), chunkandpostag(-1), chunkandpostag(-2) chunkandpostag(1), chunkandpostag(2)
	chunkandpostag$(m,n)^*$	chunkandpostag(m) & chunkandpostag(n)	chunkandpostag$(-1,0)$, chunkandpostag$(0,1)$
	default	Returns true for any context.	
BUILD	cons(n)	The head word, constituent (or POS) label, and start/join annotation of the nth tree. Start/join annotation omitted if $n \geq 0$.	cons(0), cons(-1), cons(-2), cons(1), cons(2)
	cons$(m,n)^*$	cons(m) & cons(n)	cons$(-1,0)$, cons$(0,1)$
	cons$(m,n,p)^\dagger$	cons(m), cons(n), & cons(p).	cons$(0,-1,-2)$, cons$(0,1,2)$, cons$(-1,0,1)$
	punctuation	The constituent we could join (1) contains a "[" and the current tree is a "]"; (2) contains a "," and the current tree is a ","; (3) spans the entire sentence and current tree is "."	bracketsmatch, iscomma, endofsentence
	default	Returns true for any context.	
CHECK	checkcons$(n)^*$	The head word, constituent (or POS) label of the nth tree, and the label of proposed constituent. *begin* and *last* are first and last child (resp.) of proposed constituent.	checkcons$(last)$, checkcons$(begin)$
	checkcons$(m,n)^*$	checkcons(m) & checkcons(n)	checkcons$(i, last)$, $begin \leq i < last$
	production	Constituent label of parent (X), and constituent or POS labels of children $(X_1 \ldots X_n)$ of proposed constituent	production=$X \rightarrow X_1 \ldots X_n$
	surround$(n)^*$	label of proposed constituent, and POS tag and word of the nth leaf to the left of the constituent, if $n < 0$, or to the right of the constituent, if $n > 0$	surround(1), surround(2), surround(-1), surround(-2)
	default	Returns true for any context.	

Table 6.3: Contextual Information Used by Probability Models (* = all possible less specific contexts are used, † = if a less specific context includes a word, it must include head word of the current tree, i.e., the 0th tree.)



The above action (Join VP) is encoded as follows (a vertical bar `|` separates information from the same subtree, while a comma `,` separates information from different subtrees. A tilde `~` denotes a constituent label, as opposed to a part-of-speech tag.):

```
Action = JoinVP

Contextual Predicates =
DEFAULT
cons(0)=~NP|man
cons(0*)=~NP
cons(-1)=StartVP|VBD|saw
cons(-1*)=StartVP|VBD
cons(-2)=StartS|~NP|I
cons(-2*)=StartS|~NP
cons(1)=IN|with
cons(1*)=IN
cons(2)=~NP|telescope
cons(2*)=~NP
cons(-1*,0*)=StartVP|VBD,~NP
cons(-1,0*)=StartVP|VBD|saw,~NP
cons(-1*,0)=StartVP|VBD|,~NP|man
cons(-1,0)=StartVP|VBD|saw,~NP|man
cons(0*,1*)=~NP,IN
cons(0,1*)=~NP|man,IN
cons(0*,1)=~NP,IN|with
cons(0,1)=~NP|man,IN|with
cons(0*,1*,2*)=~NP,IN,~NP
cons(0,1*,2*)=~NP|man,IN,~NP
cons(0,1,2*)=~NP|man,IN|with,~NP
cons(0,1*,2)=~NP|man,IN,~NP|telescope
cons(0,1,2)=~NP|man,IN|with,~NP|telescope
cons(-1*,0*,1*)=StartVP|VBD,~NP,IN
cons(-1*,0,1*)=StartVP|VBD,~NP|man,IN
cons(-1,0,1*)=StartVP|VBD|saw,~NP|man,IN
cons(-1*,0,1)=StartVP|VBD,~NP|man,IN|with
cons(-1,0,1)=StartVP|VBD|saw,~NP|man,IN|with
cons(-2*,-1*,0*)=StartS|~NP,StartVP|VBD,~NP|
cons(-2*,-1*,0)=StartS|~NP,StartVP|VBD,~NP|man
cons(-2,-1*,0)=StartS|~NP|I,StartVP|VBD,~NP|man
cons(-2*,-1,0)=StartS|~NP,StartVP|VBD|saw,~NP|man
cons(-2,-1,0)=StartS|~NP|I,StartVP|VBD|saw,~NP|man
```

Figure 6.9: Encoding a derivation with contextual predicates

Scoring Parse Trees

Once the probability models are estimated, we can use them to define a function score, which the search procedure uses to rank derivations of incomplete and complete parse trees. For notational convenience, define q as follows

$$q(a|b) = \begin{cases} p_{\text{TAG}}(a|b) & \text{if } a \text{ is an action from } \textsc{tag} \\ p_{\text{CHUNK}}(a|b) & \text{if } a \text{ is an action from } \textsc{chunk} \\ p_{\text{BUILD}}(a|b) & \text{if } a \text{ is an action from } \textsc{build} \\ p_{\text{CHECK}}(a|b) & \text{if } a \text{ is an action from } \textsc{check} \end{cases}$$

Let $\mathsf{deriv}(T) = \{a_1, \ldots, a_n\}$ be the derivation of a parse T, where T is not necessarily complete, and where each a_i is an action of some tree-building procedure. By design, the tree-building procedures guarantee that $\{a_1, \ldots, a_n\}$ is the only derivation for the parse T. Then the score of T is merely the product of the conditional probabilities of the individual actions in its derivation:

$$\mathsf{score}(T) = \prod_{a_i \in \mathsf{deriv}(T)} q(a_i|b_i)$$

where b_i is the context in which a_i was decided.

6.3.3 Search

The search heuristic attempts to find the best parse T^*, defined as:

$$T^* = \arg \max_{T \in \mathsf{trees}(S)} \mathsf{score}(T)$$

where $\mathsf{trees}(S)$ are all the complete parses for an input sentence S.

The heuristic employs a breadth-first search (BFS), similar to the one used in Chapter 5, which does not explore the entire frontier, but instead, explores only at most the top K scoring incomplete parses in the frontier, and terminates when it has found M complete parses, or when all the hypotheses have been exhausted. Furthermore, if $\{a_1 \ldots a_n\}$ are the possible actions for a given procedure on a derivation with context b, and they are sorted in decreasing order according to $q(a_i|b)$, we only consider exploring those actions $\{a_1 \ldots a_m\}$ that hold most of the probability mass, where m is defined as follows:

$$m = \max_m \sum_{i=1}^{m} q(a_i|b) < Q$$

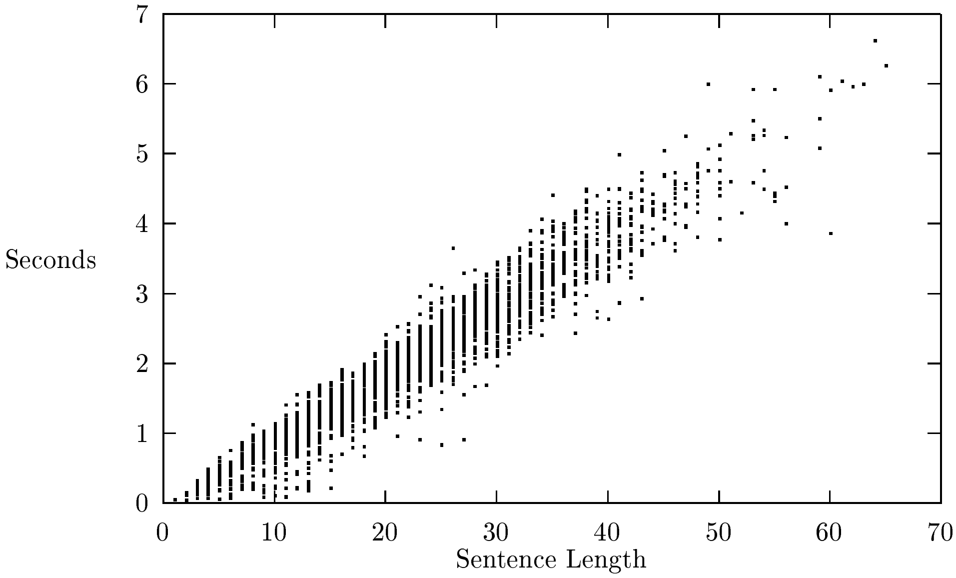


Figure 6.10: Observed running time of top K BFS on Section 23 of Penn Treebank WSJ, using one 167Mhz UltraSPARC processor and 256MB RAM of a Sun Ultra Enterprise 4000.

and where Q is a threshold less than 1. The search also uses a *Tag Dictionary* constructed from training data, described in Chapter 5, that reduces the number of actions explored by the tagging model. Thus there are three parameters for the search heuristic, namely K,M, and Q and all experiments reported in this chapter use $K = 20$, $M = 20$, and $Q = .95$[1] Table 6.4 describes the top K BFS and the semantics of the supporting functions.

It should be emphasized that if $K > 1$, the parser does not commit to a single POS or chunk assignment for the input sentence before building constituent structure. All three of the passes described in section 6.3.1 are integrated in the search, i.e., when parsing a test sentence, the input to the second pass consists of K of the best distinct POS tag assignments for the input sentence. Likewise, the input to the third pass consists of K of the best distinct chunk and POS tag assignments for the input sentence.

The top K BFS described above exploits the observed property that the individual steps of correct derivations tend to have high probabilities, and thus avoids searching a

[1]The parameters K,M, and Q were optimized on a "development set" which is separate from the training and test sets.

`advance:`	$d \times Q \longrightarrow d_1 \dots d_m$	`/* Applies relevant tree building procedure to` d `and returns list of new derivations whose action probabilities pass the threshold` Q `*/`
`insert:`	$d \times h \longrightarrow$ `void`	`/* inserts` d `in heap` h `*/`
`extract:`	$h \longrightarrow d$	`/* removes and returns derivation in` h `with highest score */`
`completed:`	$d \longrightarrow$ {`true,false`}	`/* returns true if and only if` d `is a complete derivation */`

```
M = 20
K = 20
Q = .95
C = <empty heap>          /* Heap of completed parses */
h_0 =<input sentence>      /* h_i contains derivations of length i */
while ( |C| < M )
      if ( ∀i, h_i is empty )
         then break
      i = max{i | h_i is non-empty}
      sz = min(K, |h_i|)
      for j = 1 to sz
          d_1 ... d_p = advance( extract(h_i), Q )
          for q = 1 to p
              if (completed(d_q))
                 then insert(d_q, C)
                 else insert(d_q, h_{i+1})
```

Table 6.4: Top K BFS Search Heuristic

K, M	Seconds/Sentence	Precision	Recall
20	2.07	87.9	87.1
15	1.58	87.7	86.9
10	1.07	87.7	86.9
7	0.76	87.4	86.6
5	0.56	87.3	86.8
3	0.35	86.1	86.1
1	0.14	82.4	83.4

Table 6.5: Speed and accuracy on 178 randomly selected unseen sentences

large fraction of the search space. Since, in practice, it only does a constant amount of work to advance each step in a derivation, and since derivation lengths are roughly proportional to the sentence length, we would expect it to run in linear observed time with respect to sentence length. Figure 6.10 confirms our assumptions about the linear observed running time. As expected, parsing accuracy degrades as K and M are reduced, but even with $K = 1$ and $M = 1$, accuracy is over 82%, as shown in Table 6.5.

6.4 Experiments

We present experiments that measure the accuracy and portability of the parser, and also measure the potential gain of re-scoring the parser's output. Experiments were conducted on a treebank that is widely used in the statistical natural language processing community, namely, the Wall St. Journal treebank (release 2) from the University of Pennsylvania[Marcus et al., 1994]. The maximum entropy parser was trained on sections 2 through 21 (roughly 40000 sentences) of the Wall St. Journal corpus, and tested on section 23 (2416 sentences) for comparison with other work. Table 6.6 describes the number of training events extracted from the Wall St. Journal corpus, the number of actions in the resulting probability models, and the number of selected features in the resulting probability models. Only the words, part-of-speech tags, constituent labels, and constituent boundaries of the Penn treebank were used for training and testing. The other annotation, such as the *function tags* that indiciate semantic properties of constituents, and the *null elements* that indicate traces and coreference, were removed for both training and testing.

Previous literature on statistical parsing has used the following measures, based on those proposed in [Black et al., 1991], for comparing a proposed parse P with the corresponding correct treebank parse T:

$$
\begin{aligned}
\text{Recall} &= \frac{\#\text{ correct constituents in } P}{\#\text{ constituents in } T} \\
\text{Precision} &= \frac{\#\text{ correct constituents in } P}{\#\text{ constituents in } P} \\
\text{CB} &= \text{Crossing Brackets, or \# of constituents in } P \\
&\quad \text{that violate at least one constituent boundary in } T \\
\text{0CB} &= \text{Zero Crossing Brackets, which is:} \\
&\quad 1 \text{ if } P \text{ contains any constituents that violate constituent boundaries in } T \\
&\quad 0 \text{ otherwise}
\end{aligned}
$$

For the Precision and Recall measures, a constituent in P is "correct" if there exists a constituent in T of the same label that spans the same words, and part-of-speech tags are not counted as constituents. The Recall, Precision, and Crossing Brackets measures are averaged across the constituents of the test set, while the 0 Crossing Brackets measure is averaged across the sentences of the test set. Table 6.10 shows results using the PARSEVAL measures, as well as results using the slightly more forgiving measures used in [Magerman, 1995].It shows that the maximum entropy parser compares favorably to other state-of-the-art systems [Magerman, 1995, Collins, 1996, Goodman, 1997, Charniak, 1997, Collins, 1997] and shows that only the results of [Collins, 1997] are better in both precision and recall. The parser of [Hermjakob and Mooney, 1997] also performs well (90% labelled precision and recall) on the Wall St. Journal domain, but uses a test set comprised of sentences with only frequent words and recovers a different form of annotation, and is therefore not comparable to the parsers in Table 6.10. Figure 6.11 shows the effects of training data size versus performance.

Procedure	Number of Training Events	Number of Actions	Number of Features
TAG	935655	43	119910
CHUNK	935655	41	230473
CHECK	1097584	2	182474
BUILD	1097584	52	532814

Table 6.6: Sizes of Training Events, Actions, and Features

Word:	The	man	buys	cars	with	big	tires
Modifies:	man	buys	ROOT	buys	cars	tires	with

Table 6.7: The parse of Figure 6.1, in dependency syntax notation

6.4.1 Dependency Evaluation

It is easier to diagnose errors with *dependency syntax* notation, as opposed to phrase structure notation. Any phrase structure tree annotated with head words, like the one in Figure 6.1, can be converted into dependency syntax notation, shown in Table 6.7, in which each word is "tagged" with the the word it modifies, and the head word of the sentence is tagged with the symbol ROOT. (E.g., see [Eisner, 1997] as an example of using the Penn treebank for dependency parsing and evaluation.) Table 6.8 shows the top 20 dependency errors of the parser when the training set (sections 2 through 21 of the WSJ treebank) itself is parsed, and Table 6.9 shows the top 20 dependency errors listed by part-of-speech tags, over the training set. Table 6.8 displays the source word, its correct target, and the (incorrect) target proposed by the parser, while Table 6.9 displays the tag of the source word, the tag of the correct target, and the tag of the (incorrect) proposed target. Even if the POS tags of the correct target and proposed target are be identical in Table 6.9, they correspond to different words in the sentence. Surprisingly, a few words, such as *is*, *ago*, and *from* are repeatedly involved in the errors, and commas as well as prepositions (words with part-of-speech tag IN) are consistently misattached by the parser.

6.4.2 Portability

Portability across domains is an important concern, since corpus-based methods will suffer in accuracy if they are tested in a domain that is unrelated to the one in which they are

Count	Source Word	Correct Target	Proposed Target
48	,	share	from
34	ago	from	year
32	year	ago	from
29	,	is	said
29	,	is	is
24	and	to	to
22	said	is	ROOT
21	,	is	says
20	.	is	said
19	is	is	ROOT
19	.	is	is
18	is	ROOT	said
18	.	to	is
18	,	said	is
16	,	share	rose
16	,	said	was
15	year	earlier	from
15	earlier	from	year
15	,	will	will
15	,	was	said

Table 6.8: The top 20 dependency errors on training set, by word

Count	Tag of Source Word	Tag of Correct Target	Tag of Proposed Target
582	,	NN	NN
579	NNP	NNP	NNP
526	IN	NN	NN
484	IN	VBD	NN
471	IN	VB	NN
464	,	NN	NNP
450	NN	NN	IN
447	,	VBD	VBD
438	NN	IN	NN
366	NNP	NN	NNP
366	IN	NN	VBD
333	,	VBD	NN
325	.	VBD	VBD
312	NN	IN	IN
305	NNP	NNP	NN
296	VBD	VBD	ROOT
289	VBD	ROOT	VBD
289	TO	VBD	NN
289	NNP	IN	NNP
288	NN	NN	NN

Table 6.9: The top 20 dependency errors on training set, by part-of-speech tag

Parser	Precision	Recall	CB	0 CB
Maximum Entropy$^\diamond$	86.8%	85.6%	1.27	58.7%
Maximum Entropy*	87.5%	86.3%	1.21	60.2%
[Magerman, 1995]*	84.3%	84.0%	1.46	54%
[Collins, 1996]*	85.7%	85.3%	1.32	57.2%
[Goodman, 1997]*	84.8%	85.3%	1.21	57.6%
[Charniak, 1997]*	86.7%	86.6%	1.2	59.5%
[Collins, 1997]*	88.1%	87.5%	1.07	63.9%

Table 6.10: Results on 2416 sentences of section 23 (0 to 100 words in length) of the WSJ Treebank. Evaluations marked with $^\diamond$ ignore quotation marks. Evaluations marked with * collapse the distinction between `ADVP` and `PRT`, and ignore *all* punctuation.

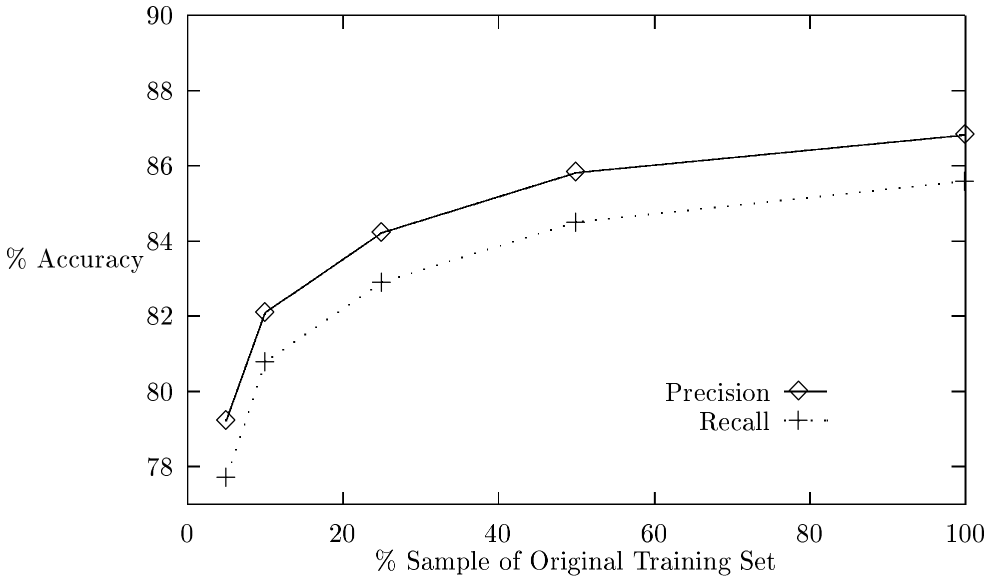


Figure 6.11: Performance on section 23 as a function of training data size. The X axis represents random samples of different sizes from sections 2 through 21 of the Wall St. Journal corpus.

Name	Description	Category
WSJ.train	Sections 2 through 21 of the WSJ corpus	Financial News
G.train	First 2000 sentences of section G in Brown corpus	Magazine/Journal Articles
G.test	Remaining 1209 sentences of section G in Brown corpus	Magazine/Journal Articles
K.train	First 2000 sentences of section K in Brown corpus	General Fiction
K.test	Remaining 2006 sentences of section K in Brown corpus	General Fiction
N.train	First 2000 sentences of section N in Brown corpus	Adventure Fiction
N.test	Remaining 2121 sentences of section N in Brown corpus	Adventure Fiction

Table 6.11: Description of training and test sets

Training Strategy	Test Corpus (Precision/Recall)			Avg. Precision/Recall
	G	K	N	
Strategy 1 : Train on WSJ.train, test on X.test	80.2%/79.5%	79.1%/78.8%	80.6%/79.9%	80.0%/79.4%
Strategy 2 : Train on WSJ.train + X.train, test on X.test	81.0%/80.5%	80.9%/80.3%	82.0%/81.0%	81.3%/80.6%
Strategy 3 : Train on X.train, test on X.test	78.2%/76.3%	77.7%/76.7%	78.7%/77.6%	78.2%/76.9%

Table 6.12: Portability Experiments on the Brown corpus. See Table 6.11 for the training and test sets.

trained (e.g., see [Sekine, 1997]). Since treebank construction is a time-consuming and expensive process, it is unlikely (in the near future) that treebanks will exist for every domain that we could conceivably want to parse. It then becomes important to quantify the potential loss in accuracy when training on a treebanked domain, like the Wall St. Journal, and testing on a new domain. The experiments here address the two following practical questions :

- How much accuracy is lost when the parser is trained on the Wall St. Journal domain, and tested on another domain (compared to when the parser is trained and tested on the Wall St. Journal) ?
- How much does a small amount of additional training material (2000 sentences) on a new domain help the parser's accuracy on the new domain ?

The new domains, namely "Magazine & Journal Articles", "General Fiction", and "Adventure Fiction", are from the Brown corpus[Francis and Kucera, 1982], a collection of English text from Brown University that represents a wide variety of different domains. These domains have been annotated in a convention similar to the text of the Wall St. Journal treebank.

Table 6.12 describes the results of several different training schemes, and table 6.11 describes the training and test corpora. The feature sets of the parser were not changed in any way when training from the Brown corpus domains. According to table 6.12, the training schemes for parsing a new domain D, ranked in order from best to worst, are:

1. Strategy 2: Train on a mixture of a lot of WSJ and a little of D
2. Strategy 1: Train on a lot of WSJ
3. Strategy 3: Train on a little of D

All experiments on a particular new domain (G, K, and N) are controlled to use the same test set, and the additional training sets G.train, K.train, and N.train all consist of 2000 sentences from their respective domain. Compared to the accuracy achieved when training and testing on the Wall St. Journal (86.8% precision/85.6% recall as shown in table 6.10), we conclude that:

- on average, we lose about 6.8% precision and 6.2% recall when training on the Wall St. Journal and testing on the Brown corpus (strategy 1),
- on average, we lose 5.5 % precision and 5% recall when training on the Wall St. Journal and the domain of interest, and testing on that same domain (strategy 2).

The discussion thus far has omitted one other possibility, namely, that the lower Brown corpus performance in strategies 1 and 2 is due to some inherent difficulty in parsing the Brown corpus text, and not to the mismatch in training and test data. A quick glance at figure 6.11 and table 6.12 dispels this possibility, since training on roughly 2000 sentences of the Wall St. Journal yields 79% precision and 78% recall, which is only slightly higher (1%) than the results on the Brown corpus under identical circumstances, roughly 78% precision 77% recall. Since the difference in accuracy due to inherent parsing difficulty (1%) is dwarfed by the loss in accuracy (7-5%) that we suffer with strategies 1 and 2, the training domain/test domain mismatch must account for most of the accuracy loss.

6.4.3 Reranking the Top N

It is often advantageous to produce the top N parses instead of just the top 1, since additional information can be used in a secondary model that re-orders the top N and hopefully improves the quality of the top ranked parse. (E.g., see [Ratnaparkhi et al., 1994b] for a probability model that reranks the output of [Jelinek et al., 1994].) Suppose there exists a "perfect" reranking scheme that, for each sentence, magically picks the *best* parse from the top N parses produced by the maximum entropy parser, where the *best* parse has the highest average precision and recall when compared to the treebank parse. The performance of this "perfect" scheme is then an upper bound on the performance of any reranking scheme that might be used to reorder the top N parses. Figure 6.12 shows that the "perfect" scheme would achieve roughly 93% precision and recall, which is a dramatic increase over the top 1 accuracy of 87% precision and 86% recall. Figure 6.13 shows that the "Exact Match", which counts the percentage of times the proposed parse P is identical (excluding POS tags) to the treebank parse T, rises substantially to about 53% from 30% when the "perfect" scheme is applied. It is not surprising that the accuracy improves by looking at the top N parses, but it is suprising—given the thousands of partial derivations that are explored and discarded—that the accuracy improves drastically by looking at *only the top 20 completed parses.* For this reason, research into reranking schemes appears to be a promising and practical step towards the goal of improving parsing accuracy.

6.5 Comparison With Previous Work

When compared to other parsers, the accuracy of the maximum entropy parser is state-of-the-art. It performs slightly better than or equal to most of the other systems compared in Table 6.10, and performs only slightly worse than [Collins, 1997]. However, the differences in accuracy are fairly small, and it is unclear if the differences will matter to the performance of applications that require parsed input. The main advantage of the maximum entropy parser is not its accuracy, but that it achieves the accuracy using only simple facts about data that have been derived from linguistically obvious intuitions about parsing. As a result, the evidence it needs can be specified concisely, and the method can be re-used

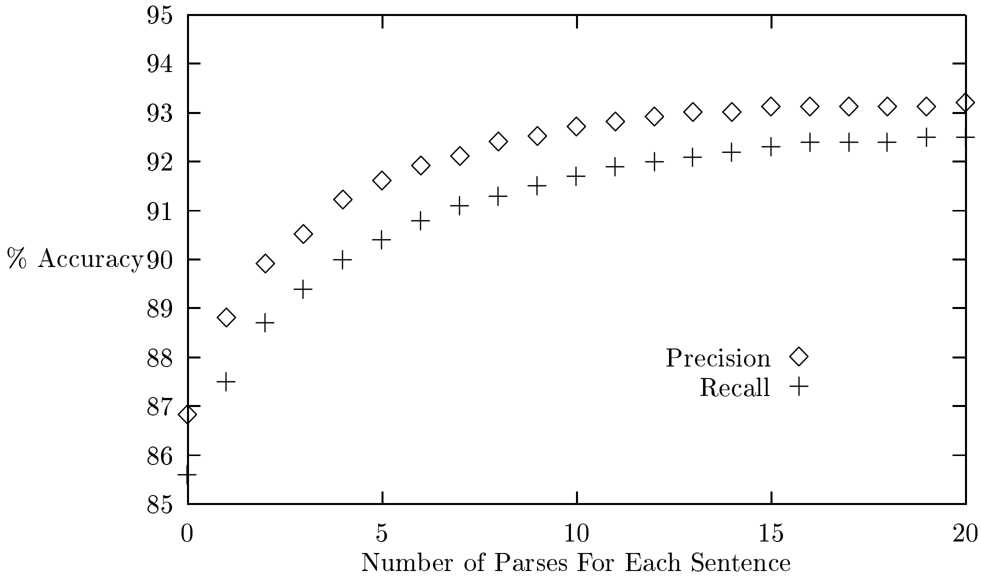


Figure 6.12: Precision & recall of a "perfect" reranking scheme for the top N parses of section 23 of the WSJ Treebank, as a function of N. Evaluation ignores quotation marks.

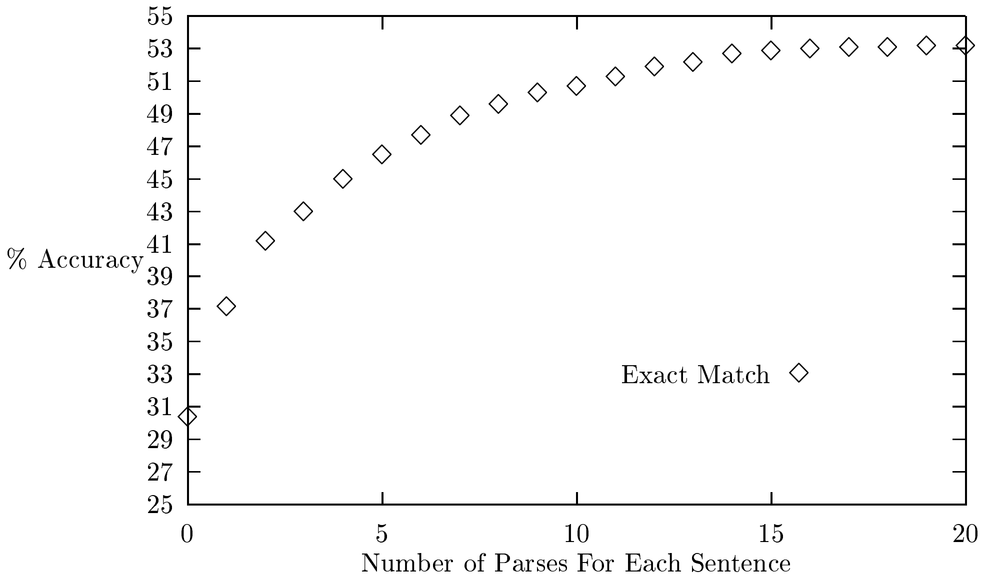


Figure 6.13: Exact match of a "perfect" reranking scheme for the top N parses of section 23 of the WSJ Treebank, as a function of N. Evaluation ignores quotation marks.

from other tasks, resulting in a minimum amount of effort on the part of the experimenter.

The maximum entropy parser differs from other statistical parsers in how it represents words and the generality of the method with which it uses to learn parsing actions. For example, the parsers of [Black et al., 1993, Jelinek et al., 1994, Magerman, 1995] use a general learning technique—decision trees—to learn parsing actions, and need to represent words as bitstrings derived from a statistical word clustering technique. The maximum entropy parser also uses a general learning technique but does not require a (typically expensive) clustering procedure, and allows experimenters to use natural linguistic representations of words and constituents.

Other parsers, like those of [Collins, 1996, Goodman, 1997, Charniak, 1997, Collins, 1997] use natural linguistic representations of words and constituents, but do not use general machine learning techniques. Instead, they use custom-built statistical models that combine evidence in clever ways to achieve high parsing accuracies. While it is always possible to tune such methods to maximize accuracy, the methods are specific to the parsing problem and require non-trivial research effort to develop. In contrast, the maximum entropy parser uses an existing modeling framework that is essentially independent of the parsing task, and saves the experimenter from designing a new, parsing-specific statistical model.

In general, more supervision typically leads to higher accuracy. For example, [Collins, 1997] uses the semantic tags in the Penn treebank while the other, slightly less accurate parsers in table 6.10 discard this information. Also, [Hermjakob and Mooney, 1997] use a hand-constructed knowledge base and subcategorization table and report 90% labelled precision and recall, using a different test set and evaluation method. Currently, the maximum entropy parser does not use this additional information, but it could, in theory, be implemented as features in the parser's appropriate probability model.

The portability of all the parsers discussed here is limited by the availability of treebanks. Currently, few treebanks exist, and constructing a new treebank requires a tremendous amount of effort. It is likely that all current corpus-based parsers will parse text less accurately if the domain of the text is not similar to the domain of the treebank that was used to train the parser.

6.6 Conclusion

The maximum entropy parser achieves state-of-the-art parsing accuracy, and minimizes the human effort necessary for its construction through its use of both a general learning technique, and a simple representation derived from a few intuitions about parsing. Those results which exceed those of the parser presented here require much more human effort in the form of additional resources or annotation. In practice, it parses a test sentence in linear time with respect to the sentence length. It can be trained from other domains without modification to the learning technique or the representation. Lastly, this paper clearly demonstrates that schemes for reranking the top 20 parses deserve research effort since they could yield vastly better accuracy results.

The high accuracy of the maximum entropy parser also has interesting implications for future applications of general machine learning techniques to parsing. It shows that the procedures and actions with which a parser builds trees can be designed independently of the learning technique, and that the learning technique can utilize the exactly same sorts of information, e.g., words, tags, and constituent labels, that might normally be used in a more traditional, non-statistical natural language parser. This implies that it is feasible to use maximum entropy models and other general learning techniques to drive the actions of other kinds of parsers trained from more linguistically sophisticated treebanks. Perhaps a better combination of learning technique, parser, and treebank will exceed the current state-of-the-art parsing accuracies.

Chapter 7

Unsupervised Prepositional Phrase Attachment

7.1 Introduction

Prepositional phrase attachment, a sub-task of the general natural language parsing problem, is the task of choosing the attachment site of a preposition that corresponds to the interpretation of the sentence. For example, the task in the following examples is to decide whether the preposition *with* modifies the preceding noun phrase (with head word *shirt*) or the preceding verb phrase (with head word *bought* or *washed*).

1. I bought the shirt with pockets.

2. I washed the shirt with soap.

In sentence 1, *with* modifies the noun *shirt*, since *with pockets* describes the *shirt*. However in sentence 2, *with* modifies the verb *washed* since *with soap* describes how the shirt is *washed*. While this form of attachment ambiguity is usually easy for people to resolve, a computer requires detailed knowledge about words (e.g., *washed* vs. *bought*) in order to successfully resolve such ambiguities and predict the correct semantic interpretation.

7.2 Previous Work

Most of the previous successful approaches to this problem have been statistical or corpus-based, and they consider only prepositions whose attachment is ambiguous between a preceding noun phrase and verb phrase. Previous work has framed the problem as a classification task, in which the goal is to predict the correct attachment $a \in \{N, V\}$, corresponding to noun or verb attachment, given the head verb v, the head noun n, the preposition p, and optionally, the object of the preposition $n2$. For example, the $(v, n, p, n2)$ tuples corresponding to the example sentences are

1. bought shirt with pockets
2. washed shirt with soap

The correct classifications of examples 1 and 2 are N and V, respectively.

[Hindle and Rooth, 1993] describes a partially supervised approach in which the Fidditch partial parser was used to extract (v, n, p) tuples from raw text, where p is a preposition whose attachment is ambiguous between the head verb v and the head noun n. The extracted tuples are then used to construct a classifier, which resolves unseen ambiguities at around 80% accuracy. Later work, such as [Ratnaparkhi et al., 1994a, Brill and Resnik, 1994, Collins and Brooks, 1995, Merlo et al., 1997, Zavrel and Daelemans, 1997, Franz, 1997], trains and tests on quintuples of the form $(v, n, p, n2, a)$ extracted from the Penn treebank[Marcus et al., 1994], and has gradually improved on this accuracy with other kinds of statistical learning methods, yielding up to 84.5% accuracy[Collins and Brooks, 1995]. [Stetina and Nagao, 1997] have reported 88% accuracy by using a corpus-based model in conjunction with a semantic dictionary, and therefore claim to match the human performance for this task reported in [Ratnaparkhi et al., 1994a]. [de Lima, 1997] uses shallow parsing techniques to collect training data for a corpus-based method to resolve ambiguous attachments in the German language.

While previous corpus-based methods approach the accuracy of humans for this task, they are not portable because they require resources that are expensive to construct or simply nonexistent in other languages. Even in English, portability to other genres is a serious hurdle for supervised approaches ; Chapter 6 shows that training a natural language parser on a treebank in one genre and testing it in another genre leads to a substantial loss in prediction accuracy. We present an unsupervised algorithm for prepositional phrase attachment that requires only a part-of-speech tagger and a morphology database during its training phase, and is therefore less resource-intensive and more portable than previous approaches, which have all required either treebanks or partial parsers. In theory, our algorithm can be easily re-trained on most genres of English, and also other languages with similar word orders. We present results in both English and Spanish.

7.3 Unsupervised Prepositional Phrase Attachment

The exact task of our algorithm will be to construct a classifier cl which maps an instance of an ambiguous prepositional phrase $(v, n, p, n2)$ to either N or V, corresponding to noun attachment or verb attachment, respectively. In the full natural language parsing task, there are more than just two potential attachment sites, but we limit our task to choosing between a verb v and a noun n so that we may compare with previous supervised attempts on this problem. While we will be given the candidate attachment sites during testing, the training procedure assumes no information about potential attachment sites.

7.3.1 Generating Training Data From Raw Text

We generate training data from raw text by using a part-of-speech tagger, a simple chunker, an extraction heuristic, and a morphology database. The order in which these tools are applied to raw text is shown in Table 7.1. The tagger from Chapter 5 first annotates sentences of raw text with a sequence of part-of-speech tags. The chunker, implemented with two small regular expressions, then replaces simple noun phrases and quantifier phrases with their head words. The extraction heuristic then finds head word tuples and their likely attachments from the tagged and chunked text. The heuristic relies on the observed fact that in English and in languages with similar word order, the attachment site of a preposition is usually located only a few words to the left of the preposition. Finally, numbers are replaced by a single token, the text is converted to lower case, and the morphology

Tool	Output
Raw Text	The professional conduct of lawyers in other jurisdictions is guided by American Bar Association rules or by state bar ethics codes , none of which permit non-lawyers to be partners in law firms .
$\Downarrow$	
POS Tagger	The_DT professional_JJ conduct_NN of_IN lawyers_NNS in_IN other_JJ jurisdictions_NNS is_VBZ guided_VBN by_IN American_NNP Bar_NNP Association_NNP rules_NNS or_CC by_IN state_NN bar_NN ethics_NNS codes_NNS ,_, none_NN of_IN which_WDT permit_VBP non-lawyers_NNS to_TO be_VB partners_NNS in_IN law_NN firms_NNS ._.
$\Downarrow$	
Chunker	conduct_NN of_IN lawyers_NNS in_IN jurisdictions_NNS is_VBZ guided_VBN by_IN rules_NNS or_CC by_IN codes_NNS ,_, none_NN of_IN which_WDT permit_VBP non-lawyers_NNS to_TO be_VB partners_NNS in_IN firms_NNS ._.
$\Downarrow$	
Extraction Heuristic	(n =lawyers, p =in, $n2$ =jurisdictions) (v =guided, p =by, $n2$ =rules)
$\Downarrow$	
Morphology:	(n =lawyer, p =in, $n2$ =jurisdiction) (v =guide, p =by, $n2$ =rule)

Table 7.1: How to obtain training data from raw text

database is used to find the base forms of the verbs and nouns.

The extracted head word tuples differ from the training data used in previous supervised attempts in an important way. In the supervised case, both of the potential sites, namely the verb v and the noun n are known in conjunction with the attachment. In the unsupervised case discussed here, the extraction heuristic only finds what it thinks are *unambiguous* cases of prepositional phrase attachment. Therefore, there is only one possible attachment site for the preposition, and either the verb v or the noun n does not exist, in the case of noun-attached preposition or a verb-attached preposition, respectively. This extraction heuristic loosely resembles a step in the bootstrapping procedure used to get training data for the classifier of [Hindle and Rooth, 1993]. In that step, unambiguous attachments from the Fidditch parser's output are initially used to resolve some of the ambiguous attachments, and the resolved cases are iteratively used to disambiguate the remaining unresolved cases. Our procedure differs critically from [Hindle and Rooth, 1993] in that we do not iterate, we extract unambiguous attachments from unparsed input sentences, and we totally ignore the ambiguous cases. It is the hypothesis of this approach that the information in *just the unambiguous* attachment events can resolve the *ambiguous* attachment events of the test data.

Tagging and Chunking

We first use the tagger in Chapter 5 to automatically annotate raw text with part-of-speech tags. Then, simple noun phrases and quantified phrases are "chunked", i.e., replaced with their head word, using the following, mostly trivial, PERL program:

```
# Input is one sentence per line,
# in the format: word1_tag1 word2_tag2 ... wordN_tagN
while (<STDIN>)
{
    # chunk simple Noun phrases, and replace with last word
    s/([^ ]+_DT )?([^ ]+_(NNP|NN|NNS|NNPS|JJ|JJS|CD) )*([^ ]+_(NNP|NN|NNS|NNPS))/$4/g;

    # chunk Quantifier phrases, and replace with last word
    s/\$_\$ ([^ ]+_CD )*([^ ]+_CD)/$2/g;
    print $_;
}
```

An example of a tagged and chunked sentence is shown in Table 7.1.

Heuristic Extraction of Unambiguous Cases

Given a tagged and chunked sentence, the extraction heuristic returns head word tuples of the form $(v, p, n2)$ or $(n, p, n2)$, where v is the verb, n is the noun, p is the preposition, $n2$ is the object of the preposition. The heuristic has the following parameters, which need to be designed by the experimenter:

The window size K: This parameter determines the maximum distance in words between a preposition p and n, v, or $n2$. We use $K = 6$ in all the experiments here.

Functions to Identify Prepositions, Nouns, and Verbs: We assume the existence of functions to identify prepositions, nouns, and verbs in tagged text.

Function to Identify Forms of *to be*: We assume a function that returns true or false to indicate if a given verb is a form of the verb *to be*.

The actual function definitions depend on the language of the text and annotation style of the tagger. Given tagged data and a description of the tagset, they are trivial to implement in English and should be easy to port to other tagsets and languages.

The main idea of the extraction heuristic is that an attachment site of a preposition is usually within a few words to the left of the preposition. We extract :

$(v, p, n2)$ if

- p is a preposition ($p \neq of$)
- v is the first verb that occurs within K words to the left of p
- v is not a form of the verb *to be*
- No noun occurs between v and p
- $n2$ is the first noun that occurs within K words to the right of p
- No verb occurs between p and $n2$

$(n, p, n2)$ if

- p is a preposition ($p \neq of$)
- n is the first noun that occurs within K words to the left of p
- No verb occurs within K words to the left of p
- $n2$ is the first noun that occurs within K words to the right of p
- No verb occurs between p and $n2$

Table 7.1 also shows the result of the applying the extraction heuristic to a sample sentence.

The heuristic ignores cases where $p = of$, since such cases are rarely ambiguous, and we opt to model them deterministically as noun attachments. We will report accuracies (in Section 7.5) on both cases where $p = of$ and where $p \neq of$. Also, the heuristic excludes examples with the verb *to be* from the training set (but not the test set) since we found them to be unreliable sources of evidence.

Morphology

We use the morphology database of the XTAG system[Karp et al., 1992] to reduce all nouns and verbs in the extracted tuples to their morphological base forms. In addition, all upper case characters are translated to lower case before using the morphology database, and any number or percent sign (%) is replaced by the token *num*. Table 7.1 shows an example in which the verb *guided* and nouns *lawyers*, *jurisdictions*, and *rules* are reduced to their base forms.

7.3.2 Accuracy of Extraction Heuristic

Applying the extraction heuristic to 970K unannotated sentences from the 1988 Wall St. Journal[1] data yields approximately 910K unique head word tuples of the form $(v, p, n2)$ or $(n, p, n2)$. The extraction heuristic is far from perfect; when applied to and compared with the annotated Wall St. Journal data of the Penn treebank, only 69% of the extracted head word tuples represent correct attachments.[2] The extracted tuples are meant to be a noisy but *abundant* substitute for the information that one might get from a treebank.

[1] This data is available from the Linguistic Data Consortium, `http://www.ldc.upenn.edu`

[2] This accuracy also excludes cases where $p = of$.

Frequency	Verb	Prep	Noun2
8110	close	at	num
1926	reach	for	comment
1539	rise	to	num
1438	compare	with	num
1072	fall	to	num
970	account	for	num
887	value	at	million
839	say	in	interview
680	compare	with	million
673	price	at	num

Table 7.2: Most frequent $(v, p, n2)$ head word tuples

Frequency	Noun	Prep	Noun2
1983	num	to	num
923	num	from	num
853	share	from	million
723	trading	on	exchange
721	num	in	num
560	num	to	month
519	share	on	revenue
461	num	to	day
417	trading	on	yesterday
376	share	on	sale

Table 7.3: Most frequent $(n, p, n2)$ head word tuples

Tables 7.2 and 7.3 list the most frequent extracted head word tuples for unambiguous verb and noun attachments, respectively. Many of the frequent noun-attached $(n, p, n2)$ tuples, such as *num to num*,[3] are incorrect. The prepositional phrase *to num* is usually attached to a verb such as *rise* or *fall* in the Wall St. Journal domain, e.g., *Profits rose 46 % to 52 million.*

[3]Recall the *num* is the token for quantifier phrases identified by the chunker, like *5 million*, or *6 %*.

7.4 Statistical Models

While the extracted tuples of the form $(n,p,n2)$ and $(v,p,n2)$ represent unambiguous noun and verb attachments in which *either* the verb or noun is known, our eventual goal is to resolve ambiguous attachments in the test data of the form $(v,n,p,n2)$, in which *both* the noun n and verb v are always known. We therefore must use any information in the unambiguous cases to resolve the ambiguous cases. A natural way is to use a classifier that compares the probability of each outcome:

$$cl(v,n,p,n2) = \begin{cases} N & \text{if } p = of \\ \arg\max_{a\in\{N,V\}} Pr(v,n,p,n2,a) & \text{otherwise} \end{cases} \tag{7.1}$$

where $Pr(v,n,p,n2,a=N)$ is the probability of the head words with a noun attachment, and where $Pr(v,n,p,n2,a=V)$ is the probability of the head words with a verb attachment.

We can factor $Pr(v,n,p,n2,a)$ as follows:

$$Pr(v,n,p,n2,a) = Pr(v)Pr(n)Pr(a|v,n)Pr(p|a,v,n)Pr(n2|p,a,v,n)$$

The terms $Pr(n)$ and $Pr(v)$ are independent of the attachment a and need not be computed in cl (7.1), but the estimation of $Pr(a|v,n)$, $Pr(p|a,v,n)$, and $Pr(n2|p,a,v,n)$ is problematic since our training data, i.e., the head words extracted from raw text, occur with either n or v, but never both n,v. This leads to make some intuitively motivated approximations for $Pr(a|v,n)$, $Pr(p|a,v,n)$, and $Pr(n2|p,a,v,n)$. Let the random variable ϕ range over $\{true, false\}$, and let it denote the presence or absence of any preposition that is unambiguously attached to the noun or verb in question. Then $p(\phi = true|n)$ is the conditional probability that a particular noun n in free text has an unambiguous prepositional phrase attachment. ($\phi = true$ will be written simply as $true$.) We approximate $Pr(a|v,n)$ as follows:

$$\begin{aligned} Pr(a=N|v,n) &\approx \frac{Pr(true|n)}{Z(v,n)} \\ Pr(a=V|v,n) &\approx \frac{Pr(true|v)}{Z(v,n)} \\ Z(v,n) &= Pr(true|n) + Pr(true|v) \end{aligned}$$

The rationale behind this approximation is that the tendency of a v, n pair towards a noun (verb) attachment is related to the tendency of the noun (verb) alone to occur with an unambiguous prepositional phrase. The $Z(v, n)$ term exists only to make the approximation a well formed probability over $a \in \{N, V\}$.

We approximate $Pr(p|a, v, n)$ as follows:

$$\begin{aligned} Pr(p|a = N, v, n) &\approx Pr(p|true, n) \\ Pr(p|a = V, v, n) &\approx Pr(p|true, v) \end{aligned}$$

and similarly approximate $Pr(n2|p, a, v, n)$:

$$\begin{aligned} Pr(n2|p, a = N, v, n) &\approx Pr(n2|p, true, n) \\ Pr(n2|p, a = V, v, n) &\approx Pr(n2|p, true, v) \end{aligned}$$

The rationale behind these approximations is that when generating p or $n2$ given a noun (verb) attachment, only the counts involving the noun (verb) are relevant, assuming also that the noun (verb) has an attached prepositional phrase, i.e., $\phi = true$. The approximations avoid using counts of n, v together, since they are never seen together in the extracted data.

We use word statistics from both the tagged corpus and the set of extracted head word tuples to estimate the probability of generating $\phi = true$, p, and $n2$. The counts used from the tagged corpus (before it has been chunked) are:

- $c(n)$: The count of a noun n
- $c(v)$: The count of a verb v

and the counts used from the extracted tuples are

- $c(n, p, n2, \phi = true)$: The count of a noun n with an unambiguously attached prepositional phrase with heads p and $n2$.
- $c(v, p, n2, \phi = true)$: The count of a verb v with an unambiguously attached prepositional phrase with heads p and $n2$.

Since the extracted tuples correspond to unambiguous attachments, they correspond to instances in which $\phi = true$. Occurrences of verbs and nouns which, according to the extraction heuristic, do not participate in unambiguous attachments correspond to instances of $\phi = false$. The relationship between the two kinds of counts is given below:

$$\begin{aligned} c(n) &= c(n, ?, ?, false) + \sum_{p,n2} c(n, p, n2, true) \\ c(v) &= c(v, ?, ?, false) + \sum_{p,n2} c(v, p, n2, true) \end{aligned}$$

where ? represents a missing head word. Other types of counts can be derived in the usual ways:

$$\begin{aligned} c(n, p, true) &= \sum_{n2} c(n, p, n2, true) \\ c(v, p, true) &= \sum_{n2} c(v, p, n2, true) \\ c(n, true) &= \sum_{p} c(n, p, true) \\ c(v, true) &= \sum_{p} c(v, p, true) \end{aligned}$$

7.4.1 Generate ϕ

The quantities $Pr(true|n)$ and $Pr(true|v)$ denote the conditional probability that n or v will occur with some unambiguously attached preposition, and are estimated as follows:

$$\begin{aligned} Pr(true|n) &= \begin{cases} \frac{c(n,true)}{c(n)} & c(n) > 0 \\ .5 & \text{otherwise} \end{cases} \\ Pr(true|v) &= \begin{cases} \frac{c(v,true)}{c(v)} & c(v) > 0 \\ .5 & \text{otherwise} \end{cases} \end{aligned}$$

7.4.2 Generate p

The terms $Pr(p|n, true)$ and $Pr(p|v, true)$ denote the conditional probability that a particular preposition p will occur as an unambiguous attachment to n or v. We present three techniques to estimate this probability, one based on raw counts, one based on an interpolation method, and one based on the maximum entropy framework.

Raw Counts

This technique uses the raw counts of the extracted head word tuples, and backs off to the uniform distribution when the denominator is zero.

$$
\begin{aligned}
Pr(p|true, n) &= \begin{cases} \frac{c(n,p,true)}{c(n,true)} & c(n, true) > 0 \\ \frac{1}{|\mathcal{P}|} & \text{otherwise, where } \mathcal{P} \text{ is the set of possible prepositions} \end{cases} \\
Pr(p|true, v) &= \begin{cases} \frac{c(v,p,true)}{c(v,true)} & c(v, true) > 0 \\ \frac{1}{|\mathcal{P}|} & \text{otherwise, where } \mathcal{P} \text{ is the set of possible prepositions} \end{cases}
\end{aligned}
$$

Interpolation

This technique is similar to the one in [Hindle and Rooth, 1993], and interpolates between the tendencies of the (v, p) and (n, p) bigrams and the tendency of the type of attachment (e.g., N or V) towards a particular preposition p. First, define c_N as the number of noun attached tuples, and $c_N(p)$ as the number of noun attached tuples with the preposition p:

$$
\begin{aligned}
c_N &= \sum_{n,p} c(n, p, true) \\
c_N(p) &= \sum_{n} c(n, p, true)
\end{aligned}
$$

Analogously, define c_V and $c_V(p)$:

$$
\begin{aligned}
c_V &= \sum_{v,p} c(v, p, true) \\
c_V(p) &= \sum_{v} c(v, p, true)
\end{aligned}
$$

Using the above notation, we can interpolate as follows:

$$
\begin{aligned}
Pr(p|true, n) &= \frac{c(n, p, true) + \frac{c_N(p)}{c_N}}{c(n, true) + 1} \\
Pr(p|true, v) &= \frac{c(v, p, true) + \frac{c_V(p)}{c_V}}{c(v, true) + 1}
\end{aligned}
$$

Interpolation via the Maximum Entropy Framework

Instead of using the above technique to interpolate between the bigrams' and attachment's tendency toward a particular preposition p, we can instead implement the (v, p) and (n, p) bigrams, as well as the $c_N(p)$ and $c_V(p)$ statistics, as features under the maximum entropy framework. Note that if we only used the (v, p) and (n, p) bigrams with a count cutoff of 0, the resulting probability model would be equivalent to the model in Section 7.4.2 that uses only the raw counts.

In the notation introduced in Chapter 2, we can define a maximum entropy conditional model q such that $q(p|n, v, a)$ is the probability of a preposition p given n, v, a, where either the noun n or verb v is unknown, depending on the value of the attachment variable a. (Here the outcome is the preposition p, and the context is the triple n, v, a) The model q is defined as follows:

Outcomes: The set of outcomes consists of the words in the data that were tagged as prepositions and that occurred at least 100 times. (The count cutoff of 100 throws away most of the tagging errors.) A special outcome `unknown` represents any preposition that is not included in the word list obtained by the frequency cutoff of 100.

Contextual Predicates: We use two predicates to capture the attachment alone:

$$\begin{aligned} cp_N(v, n, a) &= \mathit{true} \text{ iff } a = N \\ cp_V(v, n, a) &= \mathit{true} \text{ iff } a = V \end{aligned}$$

We also use two types of predicates to capture the attachment and the noun or verb, given by the following two templates:

$$\begin{aligned} cp_{noun,N}(v, n, a) &= \mathit{true} \text{ iff } a = N \text{ and } n = noun \\ cp_{verb,V}(v, n, a) &= \mathit{true} \text{ iff } a = V \text{ and } v = verb \end{aligned}$$

where *noun* and *verb* represent a noun or verb, respectively. The actual predicates are obtained automatically by matching the templates to instances in the training data. E.g., an actual contextual predicate could be:

$$cp_{buy,V}(v,n,a) = \ true \text{ iff } a = V \text{ and } v = buy$$

Recall that if $a = V$, the noun is unknown, and if $a = N$, the verb is unknown since our training data consists of only unambiguous attachments.

Feature Selection: We discard features that occur less than 5 times.

An example feature under this model might be:

$$f_{p,v,n,a} = \begin{cases} 1 & \text{if } p = with \text{ and } cp_{buy,V}(v,n,a) = true \\ 0 & \text{otherwise} \end{cases}$$

where p is outcome, and v, n, a is the context. Once q is estimated, we can compute $Pr(p|\cdot)$:

$$\begin{aligned} Pr(p|n, true) &= q(p|v = ?, n, a = Noun) \\ Pr(p|v, true) &= q(p|v, n = ?, a = Verb) \end{aligned}$$

The condition that $\phi = true$ is not written explicitly in q, but is assumed since q is trained from the unambiguous examples which represent $\phi = true$.

7.4.3 Generate $n2$

The quantities $Pr(n2|p, n, true)$ and $Pr(n2|p, v, true)$ denote the conditional probability that a noun $n2$ will occur with a preposition p and noun n, or a preposition p and verb v. Our attempts so far to use these quantities have not helped the accuracy of the classifier, so we therefore omit them in the calculation of the classifier (7.1). (Equivalently, we can assume the uniform distribution for both terms, and factor out both terms when comparing probabilities in (7.1)).

7.5 Experiments in English

Approximately 970K unannotated sentences from the 1988 Wall St. Journal were processed in a manner identical to the example sentence in Table 7.1. The result was approximately

Subset	Number of Events	$cl_{rawcount}$	cl_{interp}	cl_{maxent}	cl_{base}
$p = of$	925	917	917	917	917
$p \neq of$	2172	1620	1618	1617	1263
Total	3097	2537	2535	2534	2180
Accuracy	-	81.91%	81.85%	81.82%	70.39%

Table 7.4: Accuracy of mostly unsupervised classifiers on English

910,000 head word tuples of the form $(v, p, n2)$ or $(n, p, n2)$. Note that while the head word tuples represent correct attachments only 69% of the time, their quantity is about 45 times greater than the quantity of data used in previous supervised approaches. The extracted data was used as training material for the four classifiers cl_{base}, cl_{interp}, cl_{maxent}, and $cl_{rawcount}$ Each classifier is constructed as follows:

cl_{base} This is the "baseline" classifier, whose accuracy will indicate the level of performance we can attain using virtually no information:

$$cl_{base}(v, n, p, n2) = \begin{cases} N & \text{if } p = of \\ V & \text{otherwise} \end{cases}$$

cl_{interp}: This classifier has the form of equation (7.1), uses the method in section 7.4.1 to generate ϕ, and the "interpolation" method in section 7.4.2 to generate p.

cl_{maxent}: This classifier has the form of equation (7.1), uses the method in section 7.4.1 to generate ϕ, and the "maximum entropy" method in section 7.4.2 to generate p.

$cl_{rawcount}$: This classifier has the form of equation (7.1), uses the method in section 7.4.1 to generate ϕ, and the "raw count" method in section 7.4.2 to generate p.

Table 7.4 shows accuracies of the classifiers on the test set of [Ratnaparkhi et al., 1994a], which is derived from the manually annotated attachments in the Penn Treebank Wall St. Journal data. The Penn Treebank is drawn from the 1989 Wall St. Journal data, so there is no possibility of overlap with our training data. Furthermore, the extraction heuristic was developed and tuned on a "development set", i.e., a set of annotated examples that did not overlap with either the test set or the training set. Figure 7.1 shows the effect of

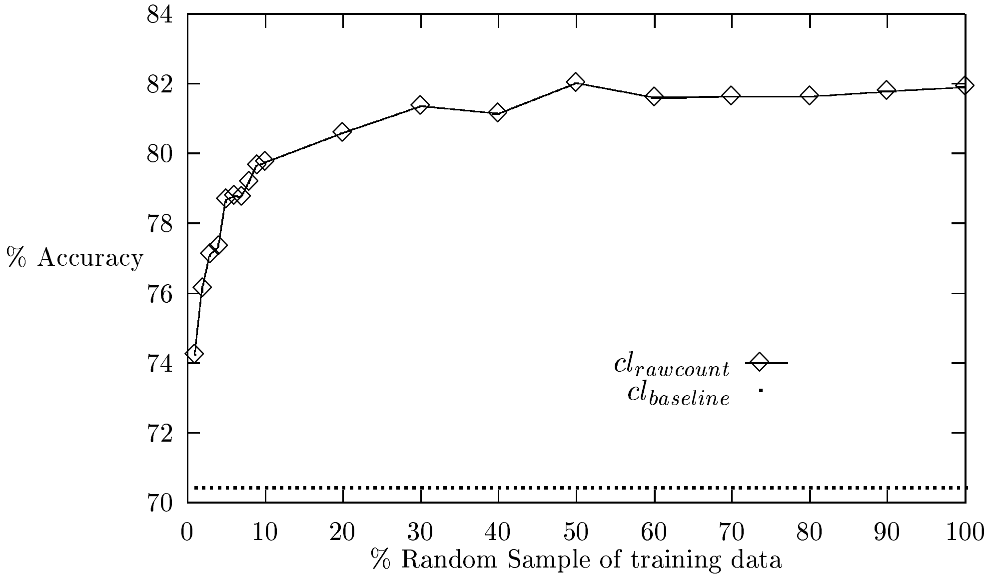


Figure 7.1: Test set performance of $cl_{rawcount}$ as a function of training set size

varying the training set size with performance on the test set, and shows that performance is unlikely to improve much with additional data.

The classifiers cl_{interp}, cl_{maxent}, and $cl_{rawcount}$ clearly outperform the baseline, and even begin to approach the performance of the best supervised approach. Surprisingly, the cl_{interp} and cl_{maxent} classifiers, which interpolate between the less specific evidence (the preposition counts) and more specific evidence (the bigram counts) do not outperform the $cl_{rawcount}$ classifier, despite the fact that they appear to be better motivated than the $cl_{rawcount}$ classifier. The failure of cl_{interp} and cl_{maxent} to outperform $cl_{rawcount}$ may be due to the errors in our extracted training data; supervised classifiers that train from clean data typically benefit greatly by combining less specific evidence with more specific evidence.

7.6 Experiments in Spanish

We claim that our approach is portable to languages with similar word order, and we support this claim by demonstrating our approach on the Spanish language. The training set of unambiguous tuples can again be extracted as in Table 7.1, but the tagger, the morphological analyzer, and the extraction heuristic must be modified for Spanish. We use the Spanish tagger and morphological analyzer developed at the Xerox Research Centre Europe[4], and we modify the extraction heuristic:

- to account for the new POS tags
- to exclude cases where the preposition is *de* or *del* (analogous to *of*)
- to correctly identify Spanish forms of *ser* (analagous to *to be*)

We did not use a chunker for the Spanish experiments, since it is more difficult to port[5] than the other natural language tools. Approximately 500k sentences of raw text from the Spanish News Text Collection were used in the Spanish experiment; the first 50k sentences were set aside to create a test set, and the remainder were used to extract the training set.

7.6.1 Creating the Test Set

Unlike English, there is no widely available test set of ambiguous prepositional phrase attachments in the Spanish language, so three annotators were hired to create such a test set. Initially, the first annotator scanned the raw text for subsequences of words $v...n...p...n2$ such that the $n2$ was the object of p, and that p was either attached to the v or the n. The annotator then recorded the head words $v, n, p, n2$ and marked them as either N or V, corresponding to either noun or verb attachment. Then, in order to shift our focus to highly ambiguous cases, the annotator extracted another test set, in which the tuples only contained the preposition *con*, since *con* was observed to be highly ambiguous in the first test set. (The test set in which $p = con$ overlaps with the first test set, but is not a subset of the first test set.)

The other two annotators were given the $(v, n, p, n2)$ head words extracted by the first annotator from both test sets and were asked to judge them as noun or verb attachments. (The judgement of the first annotator was withheld from the second and third annotators.)

[4]These were supplied by Dr. Lauri Kartunnen during his visit to Penn.

[5]It is difficult for the *author* to write a Spanish chunker. A native Spanish speaker, on the hand, would probably not find it difficult to write a chunker analagous to the one used for the English experiment.

Test Set	Subset	Number of Events	$cl_{rawcount}$	cl_{interp}	cl_{base}
All p	$p = de\|\|del$	156	154	154	154
	$p \neq de\|\|del$	116	103	97	91
	Total	272	257	251	245
	Accuracy	-	94.5%	92.3%	90.1%
$p = con$	**Total**	192	166	160	151
	Accuracy	-	86.4%	83.3%	78.6%

Table 7.5: Accuracy of mostly unsupervised classifiers on Spanish

	$cl_{rawcount}$ correct	$cl_{rawcount}$ incorrect
cl_{interp} correct	86	17
cl_{interp} incorrect	5	8

Table 7.6: Proportions correct and incorrect on Spanish data (all prepositions)

For both the test sets, the examples on which all three annotators agreed were used to evaluate the performance of our classifier on Spanish.

7.6.2 Performance on Spanish Data

The performance of the classifiers $cl_{rawcount}$, cl_{interp}, and cl_{base}, when trained and tested on Spanish language data, are shown in Table 7.5. The performance of cl_{base} is much higher in Spanish than in English, but for both Spanish test sets, the performance of $cl_{rawcount}$ exceeds that of cl_{base}.

Although the test sets for Spanish are fairly small compared to the set used in the English experiment, the difference in the performance of $cl_{rawcount}$ and cl_{base} is statistically significant. We use the significance test for non-independent proportions suggested in [McNemar, 1969], which compares the number of decisions (A) on which new classifier

	$cl_{rawcount}$ correct	$cl_{rawcount}$ incorrect
cl_{interp} correct	136	30
cl_{interp} incorrect	15	11

Table 7.7: Proportions correct and incorrect on Spanish data ($p = con$)

Attachment	$Pr(a\|v,n)$	$Pr(p\|a,v,n)$
Noun$(a = N)$	.02	.24
Verb$(a = V)$	.30	.44

Table 7.8: The key probabilities for the ambiguous example *rise num to num*

improves performance over the old classifier, with the number of decisions (B) on which the new classifier dis-improves performance. If $cl_{rawcount}$ and cl_{base} are equally accurate, we would expect that $\frac{A}{A+B} = .5$ by mere chance. We can test the null hypothesis that

$$H_0 : \frac{A}{A+B} = .5$$

versus the (two-sided) alternative

$$H_1 : \frac{A}{A+B} \neq .5$$

by assuming that the $A+B$ decisions that have changed are actually $A+B$ Bernoulli trials with A successes. Using the standard normal approximation to $A+B$ Bernoulli trials, we can reject H_0 at confidence level α if

$$\frac{A-B}{\sqrt{A+B}} \leq -z_{\alpha/2}$$

or if

$$\frac{A-B}{\sqrt{A+B}} \geq +z_{\alpha/2}$$

(See [Larsen and Marx, 1986] and [McNemar, 1969] for details on how to derive the significance test.) For the experiment in Table 7.6, $\frac{A-B}{\sqrt{A+B}} = \frac{(17-5)}{\sqrt{17+5}} = 2.56 \geq z_{.025} = 1.96$, and for the experiment in Table 7.7, $\frac{A-B}{\sqrt{A+B}} = \frac{(30-15)}{\sqrt{30+15}} = 2.23 \geq z_{.025} = 1.96$. Assuming that a $\alpha = .05$ significance level is sufficient, we can safely reject the null hypothesis (that there is no difference between cl_{base} and $cl_{rawcount}$) for both Spanish prepositional phrase attachment experiments.

7.7 Discussion

Despite the errors in the extracted head word tuples, the best performance of our unsupervised classifiers for English (81.9%) begins to approach the best performance of the comparable supervised classifiers (84.5%) in the literature. For example, Table 7.8 shows that the erroneous noun-attached head word tuple (num, to, num) is more frequent than the verb-attached (rise, to, num), but the conditional probabilities lead us to prefer the verb attachment. A comparison to the more accurate results of [Stetina and Nagao, 1997] is not useful, since our stated goal is to cheaply replicate the information in treebank, and not a semantic dictionary. In Spanish, the unsupervised classifier performs significantly better than the baseline as well, and demonstrates that our approach is inherently portable. Our results show that the information in imperfect but abundant data from unambiguous attachments, as shown in Tables 7.2 and 7.3, is sufficient to resolve ambiguous prepositional phrase attachments at accuracies just under the best comparable supervised accuracy.

There are several future directions we might take to further improve the prediction accuracy. Firstly, it may be possible to improve the extraction heuristic in a way that increases its precision but maintains its simplicity and portability. Secondly, our approach should also use the preposition $n2$, since most previous supervised approaches have used it and found that it helps accuracy (e.g., see [Brill and Resnik, 1994]). And lastly, the Spanish experiment should include a chunker, since it will allow the extraction of cleaner head word tuples. We believe that using a more precise extraction heuristic, the noun $n2$, and a chunker for Spanish will further improve the accuracies of our unsupervised approach.

The bigram-based model to compute $Pr(p|\cdot)$ for the best classifier does not fully exploit the power of the maximum entropy framework since all the features it uses—word bigrams—are homogenous, and not diverse. In circumstances such as these, maximum entropy models can be implemented with raw counts alone. (See Section 2.7.1.) However, the framework was useful for evaluating other, more diverse, feature sets in Section 7.4.2, and should be useful in future work for testing the diverse forms of information involving the second noun $n2$.

7.8 Conclusion

The unsupervised algorithm for prepositional phrase attachment presented here is the only algorithm in the published literature that can significantly outperform the baseline without using data derived from a treebank or parser. The accuracy of our technique approaches the accuracy of the best comparable supervised methods, and does so with only a tiny fraction of the supervision. Since only a small part of the extraction heuristic is specific to English, and since part-of-speech taggers and morphology databases are widely available in other languages, our approach is far more portable than previous approaches for this problem. Furthermore, we demonstrated the portability of our approach by successfully applying it to the prepositional phrase attachment task in the Spanish language.

7.9 Acknowledgments

I thank Carmen Rio-Rey, the initial annotator for the Spanish data, for the extra effort she gave in finding consistent ways to annotate the difficult cases, and also for the advice she gave me for dealing with prepositional phrases in Spanish.

Chapter 8

Experimental Comparison with Feature Selection and Decision Tree Learning

8.1 Introduction

This chapter describes controlled experiments in which we compare the maximum entropy framework— as it has been used in previous chapters— with the two following alternative modeling techniques:

Maximum Entropy Models with Incremental Feature Selection: We compare against maximum entropy probability models in which the feature set has been obtained with *incremental feature selection*, instead of a count cutoff.

Decision Trees: We also compare against the C5.0 decision tree package, which is a well known implementation of a decision tree learning algorithm.

We conduct our studies on the previously studied tasks of supervised prepositional phrase attachment and supervised text categorization.

8.2 Maximum Entropy with Feature Selection

Feature selection is the process by which we find some informative subset of features $\mathcal{F}$, given a set of pre-defined candidate features $\mathcal{C}$, where $\mathcal{F} \subseteq \mathcal{C}$. The applications in this thesis have all used a simple frequency-based count cutoff to find $\mathcal{F}$ given $\mathcal{C}$, i.e.,

$$\mathcal{F} = \{f \mid \sum_{(a,b)\in\mathcal{T}} f(a,b) \geq K \quad f \in \mathcal{C}\}$$

where $\mathcal{T} = \{(a_1, b_1) \ldots (a_N, b_N)\}$ is a training sample, K is some heuristically set threshold (usually 5 or 10), and $f(a,b) \in \{0,1\}$ is a feature. While the previous chapters show that this strategy works well in practice, the resulting set of selected features $\mathcal{F}$ is not *minimal*, in the sense that there exist features in $\mathcal{F}$ that do not contribute towards modeling the data, because they are redundant or non-informative.

There are more sophisticated strategies in the literature[Berger et al., 1996, Della Pietra et al., 1997] which incrementally attempt to build a minimal feature set $\mathcal{F}$ from a set of candidate features $\mathcal{C}$, where each feature from $\mathcal{C}$ is evaluated for its contribution to modeling the data before it is added to $\mathcal{F}$. Informally, the *incremental feature selection* (IFS) algorithm works as follows:

1. Set $\mathcal{F}_0 \leftarrow \emptyset$, set $i \leftarrow 0$, and set $\mathcal{C}$ to be the set of candidate features.
2. Select the $f \in \mathcal{C}$ that leads to the most improvement when it is added to $\mathcal{F}_i$.
3. Set $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup f$
4. Let $i \leftarrow i+1$, and if the stopping conditions are met, terminate the loop. Otherwise, repeat from (2)

Define $Q_{\mathcal{F}}$ as the set of log-linear models that use the feature set $\mathcal{F}$:

$$Q_{\mathcal{F}} = \{p | p(a|b) = \frac{1}{Z(b)} \prod_{f_j \in \mathcal{F}} \alpha_j^{f_j(a,b)}\}$$

and define $p_{\mathcal{F}}$ as the maximum likelihood model of this form:

$$p_{\mathcal{F}} = \arg\max_{p \in Q_{\mathcal{F}}} L(p)$$

At first glance, a natural way to carry out step (2) is to compute a maximum likelihood model $p_{\mathcal{F}\cup f}$ that uses the feature set $\mathcal{F} \cup f$, for every $f \in \mathcal{C}$, and to select the f for which $L(p_{\mathcal{F}\cup f}) - L(p_{\mathcal{F}})$ is the greatest. In practice, $\mathcal{C}$ is very large and the computation of $p_{\mathcal{F}\cup f}$ for every $f \in \mathcal{C}$ is very time-consuming. We therefore must approximate the contribution of each $f \in \mathcal{C}$ in a less expensive manner. Define the model form $R_{\mathcal{F},f}$ as the set of one-parameter models in which the weights of the features of $\mathcal{F}$ are fixed, but in which the weight of the candidate feature f is a parameter α:

$$R_{\mathcal{F},f} = \begin{cases} p & \mid p(a|b) = \frac{1}{Z(b)} p_{\mathcal{F}}(a|b)\alpha^{f(a,b)} \\ & \text{where } Z(b) = \sum_a p_{\mathcal{F}}(a|b)\alpha^{f(a,b)} \end{cases}$$

The maximum likelihood model of this form, $q_{\mathcal{F},f}$,

$$q_{\mathcal{F},f} = \arg\max_{p \in R_{\mathcal{F},f}} L(p)$$

is meant to be a one-parameter approximation[1] for the $|\mathcal{F}| + 1$-parameter model $p_{\mathcal{F}\cup f}$, and the quantity $L(q_{\mathcal{F},f}) - L(p_{\mathcal{F}})$ is meant to approximate the true likelihood gain of f, $L(p_{\mathcal{F}\cup f}) - L(p_{\mathcal{F}})$.

Step (4) terminates the loop if certain stopping conditions met. A good stopping condition is important, since performance will degrade on test data if either the feature set $\mathcal{F}$ is too small or too large. If $\mathcal{F}$ is too small, it will not contain all the information necessary to successfully model the data, while if it is too large, it will "overfit" the training data and perform poorly on unseen test data. A reasonable stopping condition might be to terminate the loop when the log-likelihood on "held-out" data begins to decrease as new features are added, since this is a good indicator that the feature set is starting to overfit. However, it may be the case that the held-out data log-likelihood decreases after adding a noisy and unreliable feature, but then increases much more after adding the next (more reliable) feature. In such a case, the stopping condition would terminate the loop prematurely.

In order to avoid terminating the loop prematurely, and also to avoid running the algorithm for too long, we first select N features with the IFS algorithm, where N is a

[1]We use GIS to find $q_{\mathcal{F},f}$, and it is technically a two-parameter model, since the GIS algorithm requires the additional "correction" feature.

heuristically set upper bound on the number of features necessary to accurately model the data. The final feature set $\mathcal{F}$ is the $\mathcal{F}_i \in \{\mathcal{F}_1 \ldots \mathcal{F}_N\}$ that yields the highest log-likelihood on held-out data. If $L'(p)$ denotes the held-out data log-likelihood according to the model p, we choose the feature set $\mathcal{F}$ such that:

$$\mathcal{F} = \arg \max_{F \in \{\mathcal{F}_1 \ldots \mathcal{F}_N\}} L'(p_F)$$

A more specific definition of the IFS algorithm is then:

1. Initialize $\mathcal{F}_0 = \emptyset$, initialize $\mathcal{C}$ to the set of candidate features, initialize N to be the maximum number of features that we select.

2. Select feature f that we think will most increase the likelihood of the training data, when it is added to $\mathcal{F}$:

$$f = \arg \max_{f \in \mathcal{C}} L(q_{\mathcal{F}_i, f}) - L(p_{\mathcal{F}_i})$$

3. Set $\mathcal{F}_{i+1} = \mathcal{F}_i \cup f$.

4. Set $i \leftarrow i + 1$. If $i = N$, terminate the loop and return

$$\mathcal{F} = \arg \max_{F \in \{F_1 \ldots \mathcal{F}_N\}} L'(p_F)$$

   where $L'(p)$ is the likelihood of the held-out data according to p. Otherwise, repeat from (2).

In the discussion that follows, we will fully specify an experiment with the IFS algorithm by using two parameters:

Candidate Feature Set: This is the set $\mathcal{C}$ from which the IFS algorithm will select features.

Maximum Number of Features: This is the number of features the IFS algorithm will select from $\mathcal{C}$, before it evaluates the resulting feature sets on held-out data.

We will assume the existence of a training set, a development set, and a test set. The training set will be used to build the feature sets $\mathcal{F}_1 \ldots \mathcal{F}_N$, the development set will be used to evaluate the feature sets $\mathcal{F}_1 \ldots \mathcal{F}_N$, and the test set will be used for reporting results.

The key property of this algorithm is that if a candidate feature f is redundant with some other $f' \in \mathcal{F}_i$, or if it is non-informative when compared to other features, it is not likely to be selected, since its approximate gain in likelihood (computed in step (2)) will be negligible. Therefore, the resulting set of features $\mathcal{F}$ will be far less numerous than the set of features obtained by using a count cutoff, since the redundant and non-informative features are absent. The goal of the experiments in this chapter is to see if the resulting smaller feature set will have better prediction accuracy than the much larger feature set obtained with the count cutoff.

8.3 Decision Tree Learning

Decision trees are a popular learning technique in the artificial intelligence literature, so we experimentally compare the maximum entropy technique with a commercially available decision tree package, C5.0[2], which is the successor to the well-known C4.5 package[Quinlan, 1992]. C5.0 uses a recursive partitioning algorithm and attempts to find informative tests on the *attributes* of the training data. Here, we assume that each training event consists of attributes $\{attr_1 \ldots attr_n\}$, and that each attribute $attr_i$ takes one of finitely many discrete values $v_{i1} \ldots v_{im}$. C5.0 also works with continuous numerical-valued attributes, but they are not relevant for our experiments.

C5.0 constructs a "tree"-shaped classifier, in which the "root" is at the top and the "leaves" are at the bottom. The internal nodes consist of tests and the leaves consist of a classification decision. A test at each node has the form

What is the value of $attr_i$?

and each branch leading down from the node corresponds to some value v_{ij}, that might serve as an answer to the test. When classifying a test event, we begin at the root node and trace a unique path to a leaf, by using the tests at the root and internal nodes and following

[2]This package can be licensed from Rulequest Research, `http://www.rulequest.com`

the branches that correspond to the answers to the tests. The classification returned by C5.0 is the classification at the leaf that corresponds to the test event.

Note that C5.0 is different than the statistical decision tree probability model of Section 3.2. The statistical decision tree discussed earlier returns a probability distribution, whereas C5.0 returns a classification decision (although, it appears to use counts at the leaves in arriving at its classification decision). Also, the statistical decision tree discussed earlier was binary-branching, whereas C5.0 is n-ary branching at any given node, where n depends on the possible outcomes of the test at the given node.

8.4 Prepositional Phrase Attachment

Recall that the task of prepositional phrase attachment is to take 4 head words $(v, n, p, n2)$ and classify them as either N or V, which corresponds to either noun or verb attachment. Many past supervised approaches have used the data of [Ratnaparkhi et al., 1994a], which has been extracted from the Penn treebank[Marcus et al., 1994] and is divided into a *training* set, a *development* set, and a *test* set. Each event in the training, development, and test sets is a 5-tuple

$$(v, n, p, n2, a)$$

where $v, n, p, n2$ are the appropriate head words, and $a \in \{N, V\}$. The experiments for this task are controlled to use the same training set for parameter estimation, feature selection, and decision tree induction. They use the development set for any additional parameter tuning, and use the test set only to report results. We try 3 experiments with the maximum entropy framework, called ME Default, ME Tuned, and ME IFS, and 2 experiments with decision trees, called DT Default and DT Tuned.

ME Default This is the maximum entropy framework as it has been used in this thesis, i.e., in conjunction with simple frequency based feature selection. In this experiment, any feature that occurs less than 5 times is discarded. The precise model is described below:

Outcomes: { N, V }

Contextual Predicates: Given a tuple of 4 head words $v, n, p, n2$, there exist contextual predicates that look at the following patterns

Head word 1-grams: (v), (n), (p)

Head word 2-grams: (v, p), (n, p), $(p, n2)$

Head word 3-grams: $(v, p, n2)$, $(n, p, n2)$, (v, n, p)

Head word 4-grams: $(v, n, p, n2)$

In addition we also use a *default* predicate that returns true for any context. For example, a 2-gram contextual predicate that looks at the pattern v, p might be:

$$cp_{rose,to}(b) = true \text{ if } v = \text{rose and } p = \text{to, where } b = (v, n, p, n2)$$

and a feature that uses $cp_{rose,to}$ might be:

$$f_{rose,to,V}(a, b) = \begin{cases} 1 & \text{if } a = V \text{ and } cp_{rose,to}(b) = true \\ 0 & \text{otherwise} \end{cases}$$

Feature Selection: Any feature that occurs less than 5 times is discarded.

ME Tuned This model has the same outcomes and contextual predicates as the ME Default model, but uses a different feature selection strategy. The count cutoffs have been experimentally tuned for different types of n-gram contextual predicates, depending on the n. In this experiment, all 3- and 4-gram features are kept, 2-gram features that occur less than 2 times are discarded, and 1-gram features that occur less than 10 times are discarded. The cutoffs of 1,1,2, and 10 were determined semi-automatically on a *development* set of examples, which is separate from the test set.

ME IFS This model also has the same outcomes and contextual predicates as the ME Default model, but uses a incremental feature selection instead of a count cutoff. The candidate feature set for the IFS algorithm consists of any feature that can be formed with the outcomes and contextual predicates of the ME Default model. I.e., it is equivalent to the feature set of the ME Default experiment *before* the count cutoff is applied. Figures 8.1 and 8.2 graph the likelihood and accuracy of the

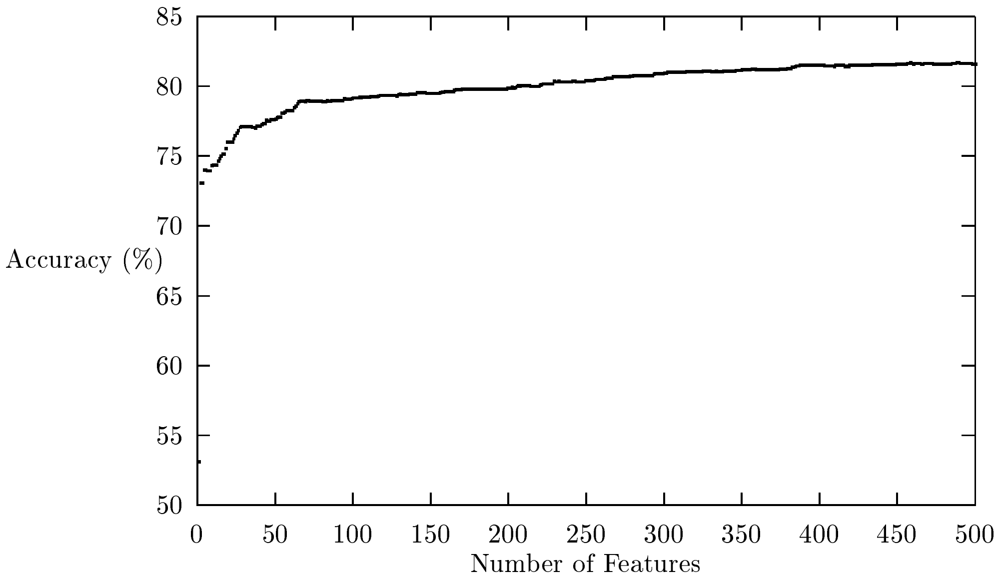


Figure 8.1: Accuracy on PP attachment development set, as features are added

development set, as a function of the number of features. Table 8.1 shows the first 20 features selected by the IFS algorithm, along with their weights. Incremental feature selection under the maximum entropy framework has been implemented before in [Ratnaparkhi et al., 1994a], but the ME IFS experiment here differs in its space of candidate features. We use a setting of $N = 500$ for the maximum number of features selected for the IFS algorithm. Ultimately, 387 features were chosen for the optimal feature set.

DT Default This is an experiment with the C5.0 package. Each training event was represented with 4 head words $v, n, p, n2$ and an answer $a \in \{N, V\}$. In the terminology used by C5.0, there are two classes $\{N, V\}$, and four attributes: verb, noun, preposition, and noun2; each attribute ranges over the corresponding words that appeared in the training data. C5.0 has 4 kinds of tests at its disposal:

- What is the value of the $verb$ attribute ?
- What is the value of the $noun$ attribute ?
- What is the value of the $preposition$ attribute ?
- What is the value of the $noun2$ attribute ?

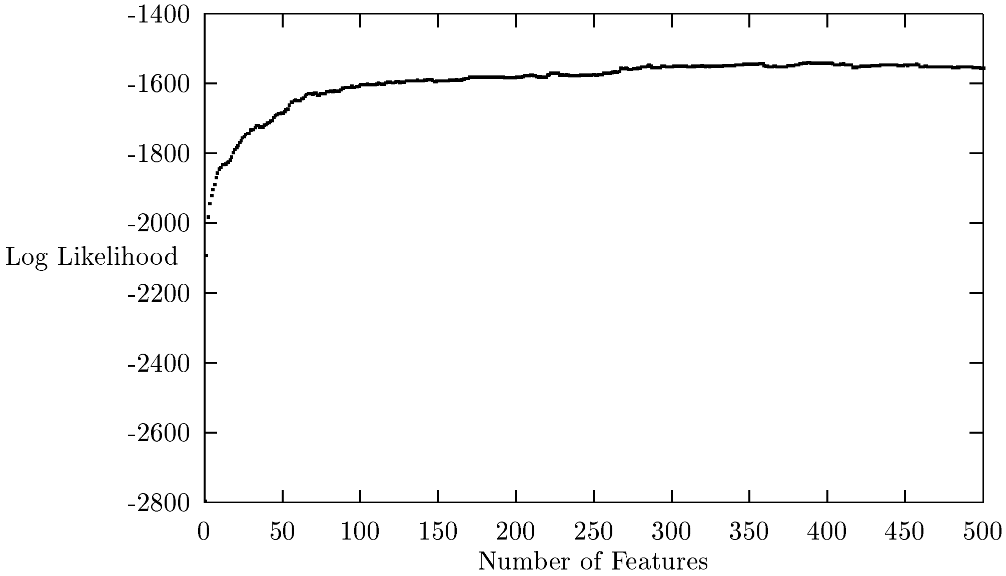


Figure 8.2: Log-Likelihood of PP attachment development set, as features are added

Feature ($cp \to a$)	Weight
p =of$\to$V	1.0e-03
DEFAULT$\to$N	8.7e-01
p =to$\to$N	1.6e-01
v =is$\to$V	2.2e-01
n =it$\to$N	3.4e-03
p =into$\to$N	8.4e-02
p =at$\to$N	2.0e-01
n =np =stake,in$\to$V	1.0e-01
$npn2$ =p =as$\to$N	2.8e-01
np =access,to$\to$V	2.0e-02
np =million,in$\to$V	6.3e-02
v =are$\to$V	2.1e-01
v =including$\to$V	1.3e-02
v =be$\to$V	3.6e-01
n =them$\to$N	8.0e-03
v =was$\to$V	2.8e-01
p =about$\to$V	3.8e-01
p =through$\to$N	1.0e-01

Table 8.1: The first 20 features selected by IFS algorithm for PP attachment

The outcomes of these questions are not binary, but rather, they are n-ary where n will be rather large. (E.g., for the first question, the number of outcomes will be the number of verbs we have seen.) The command used to invoke C5.0 is `c5.0 -f pp` where `pp` is the filestem used during testing.

DT Tuned This is also an experiment with the C5.0 package, but some of the optional parameters to C5.0 have been optimized over the development set. We report results with `c5.0 -f pp -m 5 -c 75`, where the `-m 5` prevents C5.0 from splitting a node of count less than 5, and where the `-c 75` is a pruning confidence level (75%) which allows C5.0 to induce bigger and more complex trees. If no parameters are given (as in the DT Default experiment), C5.0 assumes `-m 2` and `-c 25`.

DT Binary This is an experiment with the C5.0 package in which the representation of a context $(v, n, p, n2)$ is identical to that used in the ME Default experiment, i.e., the decision tree has access to the same contextual predicates used in the ME Default experiment. The resulting trees will be binary, since the contextual predicates are all binary-valued.

Baseline This is the performance we obtain with the following classifier:

$$cl(v, n, p, n2) = \begin{cases} N & \text{if } p = of \\ V & \text{otherwise} \end{cases}$$

The preposition *of* occurs very frequently in the data, and is almost always attached to the noun.

The results of the above experiments, the number of features in the resulting maximum entropy models, as well as the training times[3] of the maximum entropy and decision tree models, are listed in Table 8.2. All the maximum entropy models were used in a classifier cl:

$$cl(b) = \begin{cases} V & \text{if } p(a|b) >= .5 \\ N & \text{otherwise} \end{cases}$$

[3]We ran all the ME experiments on a 167Mhz UltraSPARC processor. We ran all the DT experiments on a 250Mhz UltraSPARC processor. We used a Java implementation for all the ME experiments.

Experiment	Accuracy	Training Time	# of Features
ME Default	82.0%	10 min	4028
ME Tuned	83.7%	10 min	83875
ME IFS	80.5%	30 hours	387
DT Default	72.2%	1 min	
DT Tuned	80.4%	10 min	
DT Binary	-	1 week +	
Baseline	70.4%		

Table 8.2: Maximum Entropy (ME) and Decision Tree (DT) Experiments on PP attachment

where the context $b = (v, n, p, n2)$. The decision trees grown by the C5.0 package do not return a probability distribution; they instead give a classification. The accuracy in Table 8.2 refers to classification accuracy, i.e., the number of times the classification proposed by the maximum entropy/decision tree models agreed with the actual annotated classification.

The ME Tuned experiment performs the best, the ME Default, ME IFS, and DT Tuned perform slightly worse, and the DT Default experiment performs much worse. The DT Binary experiment did not finish (even after a week of computation), so we cannot compare its accuracy with the other experiments.

The ME Default, ME Tuned, and ME IFS experiments vary only in their feature selection algorithm. The ME IFS experiment tests if incrementally selecting features from a set $\mathcal{C}$ is better than using a default (ME Default) and tuned count cutoff (ME Tuned) from the same set $\mathcal{C}$. The ME IFS experiment took much longer to run, but yields a feature set that is at least an order of magnitude less numerous than either of the feature sets selected by the ME Default and ME Tuned experiments.

The DT Default, DT Tuned, and DT Binary experiments compare the performance of decision trees against the performance of maximum entropy models (ME Default), and attempt to hold the other factor—the representation—fixed. However the representations used in DT Default and DT Tuned are slightly different, since the questions in those experiments have multiple-valued outcomes, while the contextual predicates used in the maximum entropy models are binary-valued. Although, the minimum splitting count of 5 used by the DT Tuned experiment resembles the count cutoff of 5 used by the ME Default experiment. The questions used by DT Binary are equivalent to those used in ME Default, but that experiment did not finish. It is apparently very important to tune the smoothing parameters of C5.0, since the DT Default experiment did not perform much better than the baseline for this task.

The decision trees of the DT Default and DT Tuned experiments have a harder task than the maximum entropy models of the ME experiments. The decision trees constructively *induce* conjunctions of questions, while the maximum entropy models are *told* what conjunctions to use. For example, in ME Default, we must tell the model to use trigram predicates of the form $v, p, n2$, whereas the decision trees in DT Default and DT Tuned must learn to use trigram predicates of the form $v, p, n2$. We can also give decision trees the same hints on what kinds of n-grams are useful, by using the contextual predicates of the ME Default experiment, but the experiment that uses these hints did not finish, presumably because the number of predicates is too large.

8.5 Text Categorization

In text categorization, the task is to examine a document d and predict zero, one, or more categories from a predefined set of categories as the topic(s) of the document. In our comparative experiments, we restrict ourselves to only one category, the `acq` category, which represents documents about "mergers and acquisitions".

Our task is to find a classifier

$$cl : \mathcal{B} \rightarrow \{true, false\}$$

which returns true if a document $b \in \mathcal{B}$ has the category `acq`. We implement the classifier cl with a maximum entropy probability model as follows:

$$cl(a, b) = \begin{cases} true & \text{if } p(true|b) > T \\ false & \text{otherwise} \end{cases}$$

where $T = .5$ for the experiments discussed here. In our notation for probability models, the set of contexts $\mathcal{B}$ now consists of the set of possible documents, and the outcomes of

p are $\mathcal{A} = \{true, false\}$. As a training set, we use the documents of the Reuters-21578 collection[4], which have been manually annotated with topic categories. The following experiments assume the existence of a training set $\mathcal{T} = \{(a_1, b_1) \ldots (a_N, b_N)\}$, in which each pair $(a, b) \in T$ consists of the document b, and annotation $a \in \{true, false\}$ which indicates if b is annotated with the `acq` category in the Reuters collection. The Reuters corpus is also annotated with the training set/test set split of [Apté et al., 1994], which we use for our experiments. The training set consists of 9603 documents and the test set consists of 3299 documents. We further split the original training set by using the first 7000 documents as a *development training* set, and using the remainder as a *development test* set. Following the standard convention of the text categorization literature, the words in all the documents have been lower-cased, reduced to their morphological base forms with the database of [Karp et al., 1992], and filtered with the 292-word stop list given in [Lewis, 1992]. All the learning algorithms were trained on the development training set, tuned (if necessary) on the development test set, and tested on the original test set.

We present two experiments on text categorization with maximum entropy models, called ME Default and ME IFS, and one experiment with decision trees, called DT.

ME Default The ME Default experiment uses the maximum entropy framework in conjunction with a count cutoff for feature selection. The maximum entropy probability model is described as follows:

Outcomes: { true, false }

Contextual Predicates: We use contextual predicates that check for presence of words in documents, that have the form:

$$cp_w(b) = \begin{cases} true & \text{if document } b \text{ contains word } w \\ false & \text{otherwise} \end{cases}$$

Note that the *frequency* of word w in document b is completely ignored. We also use a *default* predicate that returns true for any context.

Feature Selection: We only select features from positive examples in the training set, i.e., from documents that have been annotated as *true*, and all features

[4]This is available from `http://www.research.att.com/~lewis` for research purposes.

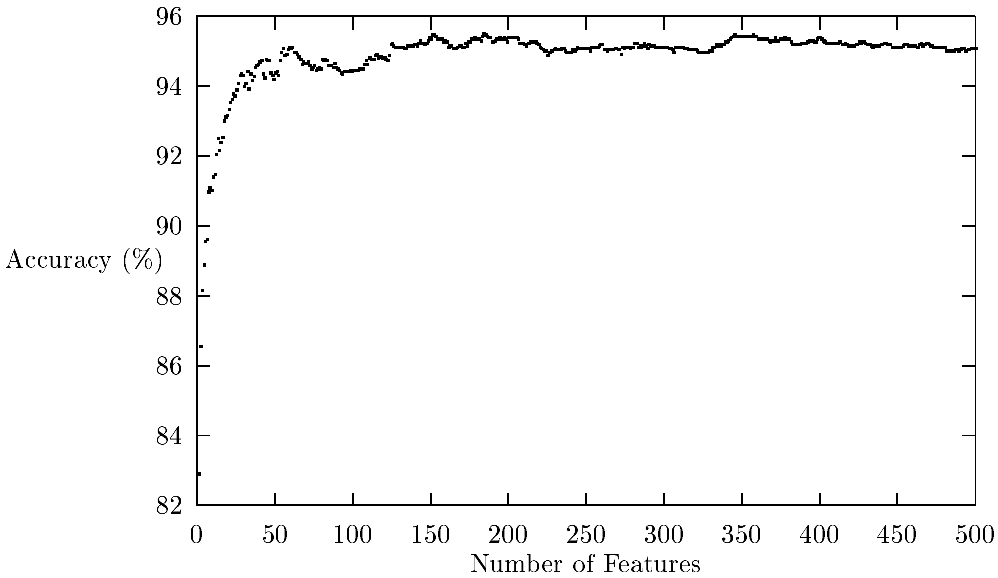


Figure 8.3: Accuracy on text categorization development set, as features are added

therefore check for $a = true$. We use all features of the form

$$f(a,b) = \begin{cases} true & \text{if } cp_w(b) = true \text{ and } a = true \\ false & \text{otherwise} \end{cases}$$

Any feature that occurs less than 5 times in the training set is discarded.

ME IFS: This experiment uses incremental feature selection with the maximum entropy framework. The outcomes and contextual predicates are the same as the ME Default experiment, but the feature set is built incrementally with the IFS algorithm. The candidate feature set consists of any feature that can be formed with the outcomes and contextual predicates of the ME Default experiment. I.e., it is equivalent to the feature set of the ME Default experiment for text categorization *before* the count cutoff is applied. Figures 8.3 and 8.4 graph the likelihood and accuracy of the development set, as a function of the number of features. Table 8.3 shows the first 20 features selected by the IFS algorithm, along with their weights. The predicate named *null* is the default predicate. We use a setting of $N = 500$ for the maximum number of features selected for the IFS algorithm. Ultimately, 356 features were chosen for the optimal feature set.

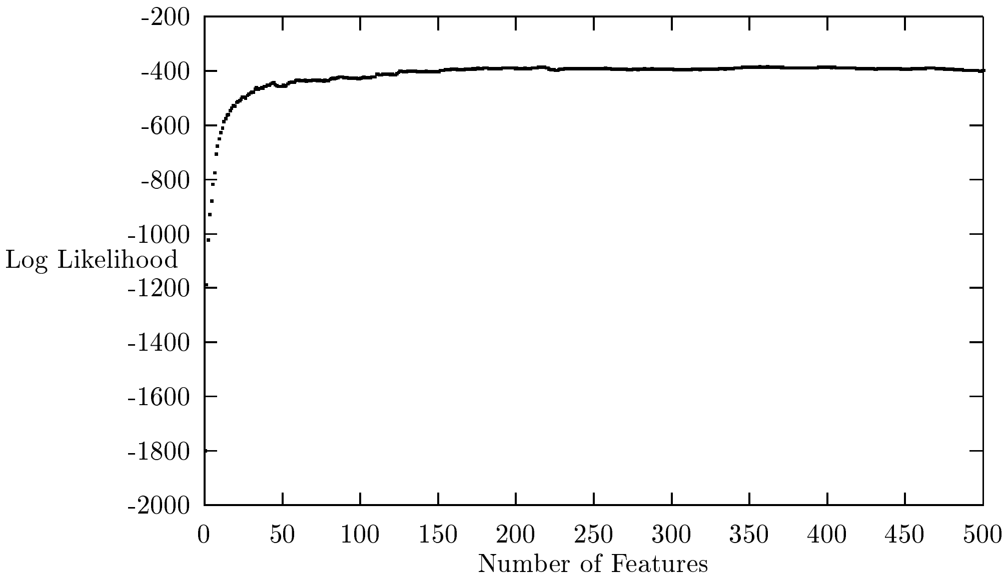


Figure 8.4: Log-Likelihood of text categorization development set, as features are added

Feature ($cp \to a$)	Weight
null→true	8.8e-01
acquire→true	1.6e+00
stake→true	1.7e+00
merger→true	1.6e+00
cts→true	2.9e-01
rate→true	3.2e-01
acquisition→true	1.6e+00
year→true	6.9e-01
share→true	1.3e+00
undisclosed→true	1.7e+00
export→true	2.7e-01
disclose→true	1.4e+00
bid→true	1.5e+00
rise→true	4.0e-01
underwrite→true	2.6e-01
unit→true	1.3e+00
debt→true	6.2e-01
tonne→true	7.8e-02
net→true	4.9e-01
sale→true	1.3e+00

Table 8.3: The 20 first features selected by IFS algorithm for text categorization

DT Default This experiment uses the C5.0 decision tree package. A document is encoded with the predicates used in the ME Default experiment (after feature selection). Given m predicates $cp_1 \dots cp_m$, we represent a document d as a m element boolean vector $\{cp_1(d) \dots cp_m(d)\}$. At each node, the decision tree effectively asks

Is word w in the document ?

and can form arbitrarily complex conjunctions of such questions. We use the command `c50 -f acq`.

DT Tuned This experiment uses the same representation as the DT experiment, but some of the optional parameters to C5.0 have been optimized over the text categorization development set. We report results with `c50 -f acq -m 5 -c 75`. which invokes C5.0 with a mininum splitting count of 5, and a pruning confidence level of 75%. (This was also the optimal setting for the prepositional phrase attachment task.)

Table 8.4 shows the accuracy and training times[5] of the ME Default, ME IFS, DT Default, and DT Tuned experiments on the text categorization task, for the `acq` category. In addition, the feature set sizes of the ME experiments are also included. The ME Default and ME IFS algorithms for text categorization vary only in their feature selection, and the results show that using incremental feature selection slightly outperforms feature selection using a count cutoff. Both ME Default and ME IFS outperform both of the DT experiments. The feature set selected by ME IFS is approximately 5 times smaller than the one selected by ME Default. The representation used in both DT experiments is exactly the same as that used in the ME Default experiment; both DT Default and DT Tuned use the contextual predicates of the selected feature set in the ME Default experiment.

The decision tree of the DT experiments has an advantage in its ability to induce conjunctions of word questions, and effectively test for word n-grams. In contrast, the maximum entropy models (as they are used here) do not induce conjunctions, they use only predicates on single words. It is surprising that this representational advantage of the DT experiments does not translate into an accuracy gain over the ME experiments.

[5]We ran all the ME experiments on a 167Mhz UltraSPARC processor. We ran all the DT experiments on a 250Mhz UltraSPARC processor. We used a Java implementation for all the ME experiments.

Experiment	Accuracy	Training Time	# of Features
ME Default	95.5%	15 min	2350
ME IFS	95.8%	15 hours	356
DT Default	91.6%%	18 hours	
DT Tuned	92.1%	10 hours	

Table 8.4: Text Categorization Performance on the `acq` category

8.6 Conclusion

The maximum entropy technique appears to perform better than the decision tree package C5.0 for both the prepositional phrase attachment task and the text categorization task. All experiments were controlled to use the same training, development, and test sets. The experiments with decision trees were controlled to use a representation that was as close as possible to the representation used by the maximum entropy models. Surprisingly, the ability of the decision tree to induce conjunctions of questions did not give it a performance advantage, since either the types of conjunctions could be pre-specified to the maximum entropy model (in the case pp attachment), or the conjunctions of questions did not appear to improve the accuracy (in the case of text categorization).

We also compared two very different feature selection strategies for maximum entropy models: count cutoff feature selection (CCFS) and incremental feature selection (IFS). The CCFS strategy performed more accurately for prepositional phrase attachment, while the IFS strategy performed more accurately for text categorization. Neither strategy performed far behind the other. CCFS has the advantage that it is extremely quick to both implement and execute, while the IFS algorithm is much more complicated and extremely time-consuming. However, the IFS algorithm yields a concise and readable list of features that represent the facts that have been learned. With CCFS, it is much more difficult to ascertain exactly what features are important for modeling the data.

The discussion suggests that if efficiency is the main objective, then CCFS is a better choice, whereas if readable features are the objective, then IFS is the choice. In terms of accuracy, neither feature selection strategy consistently outperforms the other.

Chapter 9

Limitations of the Maximum Entropy Framework

Probability models estimated under the maximum entropy framework perform well in practice, but have certain limitations that may lead to poor prediction accuracy, or may prevent the framework from capturing relevant facts about the data. The major limitation is that the exact maximum likelihood/maximum entropy solution does not exist under certain circumstances, in which case, the probability distribution resulting from the GIS algorithm may lead to poor prediction accuracy. Another, somewhat minor, disadvantage is that some facts about natural language may not be expressible through *binary-valued* features, and cannot therefore be represented in our current implementation of the maximum entropy framework. We discuss how these limitations are handled in practice.

9.1 Convergence Problems

In order to satisfy a constraint on feature expectations, the training data may require the solution to achieve $p(a|b) = 1$ for some a, b pair. In such cases, the exact solution does not exist in the form of the probability model (2.1), and the model parameters will not converge under the GIS algorithm. Furthermore, the probability estimates from the resulting model are not what one typically desires from a learning technique. We first give examples where the parameters both converge and diverge, and describe how their interaction leads to undesirable results. We also discuss other smoothing methods in the literature for dealing with this problem.

9.1.1 Exact Solution Exists: Parameters Converge

Suppose that our space of predictions is $\mathcal{A} = \{0, 1\}$, and our space of contexts is limited to $\mathcal{B} = \{x\}$. Our task is to observe some context in $\mathcal{B}$ (which trivially consists of one element), and predict the probability of seeing it with $a \in \{0, 1\}$. Assume that we are given two features f_{x0} and f_{x1}:

$$f_{x0}(a, b) = \begin{cases} 1 & \text{if } a = 0 \text{ and } b = x \\ 0 & \text{otherwise} \end{cases}$$

and

$$f_{x1}(a, b) = \begin{cases} 1 & \text{if } a = 1 \text{ and } b = x \\ 0 & \text{otherwise} \end{cases}$$

and assume that we are given the following training sample $\mathcal{T} = \{(0, x), (1, x), (1, x), (1, x), (1, x)\}$. The features do not overlap, and by using equation 2.6, we see that the maximum entropy probability model, over the constraints $E_p f_{x0} = .2$ and $E_p f_{x1} = .8$, will yield $p(0|x) = .2$ and $p(1|x) = .8$. This trivial case represents what is usually encountered in the natural language processing tasks, namely, some context x that is ambiguous between two or more outcomes, in this case $\{0, 1\}$.

9.1.2 Exact Solution does not exist: Parameters Diverge

Again suppose that $\mathcal{A} = \{0, 1\}$, and that $\mathcal{B} = \{x\}$, and suppose that our training sample is $\mathcal{T} = \{(0, x)\}$. Assume that we are given the one feature f_{x0}, as defined above. In order to meet the constraint $E_p f_{x0} = 1$, it must be the case that $p(0|x) = 1$. However, when we expand the probability model and try to explicitly calculate $p(0|x)$, we get

$$p(0|x) = \frac{\alpha_{x0}}{\alpha_{x0} + 1}$$

Clearly, the above formula cannot achieve $p(0|x) = 1$ exactly with finite values for α_{x0}. Because the model's expectation $E_p f_{x0}$ will *always* be under 1, the GIS algorithm will always increase the value of α_{x0} on each iteration, and will drive the value of $p(0|x)$ closer to 1, but will never actually reach it. The reason for the divergence of α_{x0} to $+\infty$ is that x is not ambiguous, it only ever appears with 0.

9.1.3 Parameter Interaction

Unambiguous contexts yield strange results when used together with ambiguous contexts. Suppose that $\mathcal{A} = \{0, 1\}$, and $\mathcal{B} = \{x, y, xy\}$, and that we are given two features f_{x1}, f_{y0}. Furthermore, suppose that our training set $\mathcal{T}$ consists of N elements

$$\mathcal{T} = \{(0, y), (1, x) \ldots (1, x), (0, x)\}$$

where N is some very large number, say 1,000,000. When we expand the probability model,

$$p(0|y) = \frac{\alpha_{y0}}{\alpha_{y0} + 1}$$

and

$$p(1|x) = \frac{\alpha_{x1}}{\alpha_{x1} + 1}$$

In order to meet the constraint $E_p f_{y0} = \frac{1}{N}$, α_{y0} will diverge to $+\infty$, but in order to meet the constraint $E_p f_{x1} = \frac{N-2}{N}$, α_{x1} will converge to some finite value. The probability for seeing the prediction 0 with both x and y, or $p(0|xy)$, is given by:

$$p(0|xy) = \frac{\alpha_{y0}}{\alpha_{y0} + \alpha_{x1}}$$

The term α_{y0} diverges to $+\infty$, and will always dominate the term α_{x1}. This property is highly non-intuitive and unattractive, since the outcome and context pair $(1, x)$ occurs $N = 1,000,000$ times in the training data, but yet the parameter based only on one occurrence of y takes precedence. In effect, the model gives infinite confidence to contexts that are not ambiguous with respect to the predictions with which they occur, regardless of their frequency.

In natural language, contexts that occur once or very infrequently are not as reliable as frequently occurring contexts. If the above model were implemented for language, where $\mathcal{A}$ represented some binary linguistic category and where $\mathcal{B}$ represented word occurrences, it would most likely have very low prediction accuracy, since it gives infinite confidence to an unreliable event. The count cutoff feature selection strategy used in this thesis allows us to ignore this problem in practice, since infrequent features, i.e., those that occur less than 5 or 10 times, are discarded. Most of the remaining frequent features are either ambiguous, and do not suffer from diverging parameter values, or tend to be reliable, and do not cause a loss in prediction accuracy. E.g., if the pair $y, 1$ occurs 100 times in the training data, and $y, 0$ never occurs, the model will place infinite confidence in the feature f_{y1}, but we are unlikely in practice to suffer a loss in prediction accuracy as a result of such confidence.

9.1.4 Smoothing

Other work has relied on smoothing techniques to cope with the situation in which the exact solution does not exist.

[Lau, 1994] reports results on experiments with the *fuzzy maximum entropy* framework[1] in which the objective is to maximize the sum of the entropy function and a penalty function, subject to linear constraints on feature expectations. The penalty function is heuristically selected to penalize deviations from unreliable (infrequently observed) constraints more than deviations from reliable (frequently observed) constraints. In effect, the constraints are "soft", so that strict equality of the model's feature expectation and the observed expectation is not required.

[Lau, 1994] also applies Good-Turing discounting to the observed feature expectation. E.g., using the above example, the model then would not meet the constraint $E_p f_{y0} = 1$, but instead would meet the constraint $E_p f_{y0} = 1 - \epsilon$, where ϵ is determined with the formula in [Katz, 1987]. However, in this approach, there is no guarantee that the constraints are consistent with each other, since they no longer represent counts drawn from training data.

Both techniques allow the maximum entropy solution to exist without requiring the probability model to achieve $p(a|b) = 1$.

9.2 Binary-Valued features

All of the features sets in this thesis consist of binary-valued features, which return 1 or 0 depending on the presence or absence of certain contextual evidence and a certain outcome.

[1] This framework was developed by Steven Della Pietra and Vincent Della Pietra at the IBM TJ Watson Research Center.

While such features are sufficient to capture information on the word or sentence level, they are not as useful for capturing information on the document level, in which word frequency— as opposed to word presence or absence—is an important source of evidence. Word frequency information can be captured in a somewhat *ad hoc* manner using quantized predicates. E.g., for our text categorization task, we might want a feature like

$$f_w(a,b) = \begin{cases} 1 & \text{if } a = \texttt{yes} \text{ and } count(w,b) >= 10 \\ 0 & \text{otherwise} \end{cases}$$

where a is an outcome, b is a document, and where $count(w,b)$ returns the frequency of word w in the document b. Alternatively, our the features can be generalized to return integer values, e.g.,

$$f_w(a,b) = \begin{cases} \text{count(w,b)} & \text{if } a = \texttt{yes} \text{ and } w \in b \\ 0 & \text{otherwise} \end{cases}$$

The limitation to binary-valued features is an implementation choice, and not an inherent property of the framework.

9.3 Conclusion

Probability models under the maximum entropy framework used in this thesis have a natural limitation in that they cannot model data that requires $p(a|b) = 1$ or $p(a|b) = 0$. In order to overcome this limitation, others have either modified the framework to allow soft constraints, or have applied smoothing techniques to the observed expectations. In our work, the features most likely to cause problems as a result of these limitations are the low-count features, and our use of a feature selection technique that discards low-count features greatly reduces the chance of adverse modeling effects. Our implementation has the drawback that it only uses binary-valued features to represent facts about the data. However, this limitation is not severe since we have not found that integer-valued features would have been useful for representing facts on the word and sentence level.

Chapter 10

Conclusion

The experiments in this thesis support our claims about accuracy, knowledge-poor features, and reusability. Table 10.1 summarizes the tasks, and describes the probability models that were implemented for each task. The tasks in Table 10.1 differ dramatically in their outcomes and contextual predicates, and there is no reason to suspect any underlying similarities between the contextual predicates for one model, such as POS tagging, and any other model, such as the parser's BUILD model. However, the same modeling technique, along with practically the same feature selection strategy, performs accurately on all of these tasks. We provide detailed arguments below that describe how the experiments in the thesis have fulfilled our claims about accuracy, knowledge-poor features, and reusability.

10.1 Accuracy

We claim that the use of maximum entropy models for a *wide variety* of natural language learning tasks gives highly accurate results. We have presented maximum entropy probability models for end-of-sentence detection, part-of-speech tagging, parsing, and prepositional phrase attachment that perform near or at the state-of-the art, without any substantial task-specific tuning. The maximum entropy probability models for prepositional phrase attachment and text categorization outperform the decision tree package C5.0 when trained and tested under identical or similar conditions. The few results that exceed those presented here either require additional linguistic resources (in the form of annotation or

Task	Outcomes ($\mathcal{A}$)	Contextual Predicates ($\mathcal{B}$)	Feature Selection
EOS Detection	`yes, no`	text before and after candidate punctuation mark	Count cutoff of 10
POS Tagging	Tagset of 45 part-of-speech categories	word, prefix, suffix, surrounding words, previous tags	Count cutoff of 10
Parsing: CHUNK model	`StartX, JoinX, Other`	± 2 words and POS tags	Count cutoff of 5
Parsing: BUILD model	`StartX, JoinX`	uni-,bi-,and tri-grams of head words of ± 2 constituents, punctuation	Count cutoff of 5
Parsing: CHECK model	`yes, no`	last head, first head, bigram of last head and other head in current constituent, surrounding words, CFG rule	Count cutoff of 5
Prepositional Phrase Attachment (unsupervised)	All possible prepositions	Bigrams of head words	Count cutoff of 0 and 5
Prepositional Phrase Attachment (supervised)	{ N,V }	1,2,3,4-grams of head words	Count cutoff of 5/Tuned count cutoff
Text Categorization	`yes, no`	Words	Count cutoff of 5

Table 10.1: Summary of maximum entropy models implemented in this thesis

linguistic databases), or require additional research on building task-specific statistical models. Furthermore, it is usually possible to incrementally increase accuracy by adding more interesting features as they are discovered.

10.2 Knowledge-Poor Feature Sets:

We claim that these high accuracies can be obtained with *knowledge-poor* feature sets. By knowledge-poor, we mean that the features do not require linguistic expertise, and that they test only for simple co-occurrences of words and linguistic material in the context. The feature sets do not rely upon manually constructed linguistic or semantic classes besides those already pre-existing in the training corpus annotation. The feature sets are knowledge-poor by design; our objective has been to impart as little knowledge as possible to the computer, and force it to learn as much as possible from the data.

While it is conceivable that knowledge-rich features could improve the accuracy, our studies have been restricted to knowledge-poor features since they are very inexpensive to implement, and can be ported to other genres and languages more easily than knowledge-rich features.

We discuss why the feature sets used in the tasks of Table 10.1 all represent advances in that they require less supervision, knowledge, or preprocessing from the researcher than competing approaches in the literature, yet perform better or comparable to the competing approaches. In section 10.2.5, we discuss why a count cutoff works well as a feature selection strategy.

10.2.1 End-of-sentence detection

Our approach for this task uses only word spellings, and optionally, an abbreviation list, whereas other approaches, such as [Palmer and Hearst, 1997], require part-of-speech tags in addition to these features. Since we do not use part-of-speech tags, our approach is far more portable to other languages than previous approaches.

10.2.2 Part-of-speech tagging

The tagger in this thesis automatically derives features for tagging unknown words from the training corpus, and uses them together with the usual features for tagging known words (e.g., the previous tag, the previous two tags) in a unified probability model. It differs from taggers like [Brill, 1994], which uses separate learners for tagging known and unknown words, and differs from [Weischedel et al., 1993], which needs manually pre-specified suffix lists to tag unknown words.

10.2.3 Parsing

The features of the parser are knowledge-poor in the sense that they are derived from four, fairly shallow, intuitions about parsing. The parser also requires less pre-processing and linguistic information than other parsers built with general learning algorithms. For example, it differs from the decision tree parsers of [Black et al., 1993, Jelinek et al., 1994, Magerman, 1995] in that it does not require preprocessing of the words into statistically-derived word classes. It also differs from the decision tree/decision list parser of [Hermjakob and Mooney, 1997] in that it does not use a hand-constructed knowledge base. And because the parser uses a general learning algorithm, its features do not need to be as carefully selected as those used in other approaches based on parsing-specific learning frameworks, such as [Collins, 1996, Goodman, 1997, Charniak, 1997, Collins, 1997].

10.2.4 Unsupervised Prepositional Phrase Attachment

The features used in this model are derived from data that has been annotated with part-of-speech tags and morphology information, but not attachment information. It differs from all previous approaches, which have all used either treebanks or parsers to obtain the relevant statistics.

10.2.5 Why Count Cutoffs Work

The feature sets for all the tasks discussed above have been obtained with count cutoff feature selection. It will be illustrative to know *why* a simple count cutoff suffices as a feature selection strategy, despite the fact that it discards information. We use a count cutoff of $K > 0$ for most of the tasks, but use $K = 0$ for the unsupervised prepositional phrase attachment. The exact count cutoff is determined by the nature of the feature set. Feature sets that use $K > 0$ consist of very specific features as well as less specific, or generalized features. In contrast, the feature set that uses $K = 0$ consists only of specific features (namely, head word bigrams). These properties suggest the conditions for how best to select K:

Use $K > 0$ when the feature set has both specific and generalized features. Most of the features discarded will be specific features, since such features will tend to have low frequencies. We discard them on the hypothesis that such features are unreliable sources of evidence. Therefore both specific and generalized features will be used to model an event only if the specific features are frequent (and hence reliable), whereas only generalized features will be used if the specific features corresponding to that event have been discarded. Our discussion has been limited to two kinds of features, "specific" and "generalized", but in practice, the feature set consists of many kinds of features of varying degrees of generality. Using a non-zero count cutoff is an easy way of automatically selecting the levels of generality that are to be included in the feature set.

The optimal cutoff K can usually be found semi-automatically, by evaluating the performance of several models on held-out data, where each model corresponds to a different value of K. For the tasks in this thesis, we have not invested much effort on finding the optimal K for each task; $K = 5$ and $K = 10$ appear to work well for a variety of tasks. However, it is possible that more careful tuning of K could further boost accuracies.

Use $K = 0$ when the feature set consists of only specific features (and no generalized features). In this case, using $K > 0$ will actually throw away valuable information, since the model cannot fall back on generalized features, if the necessary specific features do not exist.

In most of our applications, there will exist a hierarchy of features of varying generality, and using $K > 0$ will be appropriate. The point of maximum entropy modeling is to combine different kinds of evidence; maximum entropy models that use homogeneous forms of evidence can be implemented with much simpler techniques, e.g., the cl_{bigram} classifier for unsupervised prepositional phrase attachment in Chapter 7.

10.3 Software Re-usability

An advantage of the maximum entropy framework is that its software implementation is highly reusable. The theory of the maximum entropy framework is independent of any one particular natural language task. In practice, since we are not using any task-specific modifications, the software developed for one task can be re-used for other tasks that are implemented under this framework. A single software implementation can train all of the maximum entropy probability models in this thesis. (We actually used two versions in this thesis; the first one was written in C++, and the second one was written in Java to increase portability to other platforms.)

10.4 Discussion

Our claims have important practical ramifications for researchers in natural language processing. Researchers can view the maximum entropy framework as a re-usable, general purpose modeling tool for any natural language problem that can be reformulated as a machine learning task. They can encode their data with knowledge-poor features, and our experiments suggest that they can expect high performance on their unseen test sets. As a consequence, researchers need only concentrate effort on the discovery of the information that is necessary to solve the problem, and need not spend time on finding specialized models that combine the information that was discovered.

However, in theory, *any* general purpose machine learning algorithm should be able to fulfill our claims of reusability, knowledge-poor features, and accuracy across tasks. However, few general learning algorithms in the computational linguistics literature have been *demonstrated* to work this accurately across tasks, and to scale up to problems as large as parsing. For example, decision trees have been applied extensively for solving problems in natural language processing, but the technique used in this thesis outperforms a popular decision tree package on both the prepositional phrase attachment and text categorization tasks. This thesis is not the first application of the maximum entropy technique to natural language, but it is the only study of the framework that has demonstrated consistently high accuracies across different tasks with the same feature selection strategy.

10.5 Future Work

In future work, we intend to apply natural language learning techniques to corpora that are linguistically deeper, as well as corpora that are not linguistically annotated.

Statistical approaches to natural language have often been criticized for their inability to deal with problems deeper than syntax and shallow semantics. We believe this criticism merely reflects the nature of the available annotated data, and not the approach itself. An interesting future direction is to annotate text with a deeper level of semantic information, in the hope that it can be learned with the application of statistical techniques and knowledge-poor features. Such corpora would enable statistical techniques to return analyses for natural language that are both semantically deeper, and hopefully more accurate than those that are returned by current statistical techniques.

Secondly, all the tasks in this thesis, excluding Chapter 6, have assumed the existence of a large, annotated training set. Linguistic annotation is expensive, and limits the portability of supervised natural language learning methods. It is therefore in the interest of the natural language processing community to develop unsupervised methods that learn linguistic information without using time-intensive linguistic annotation, so that natural language problems can be solved for other genres of English, and other languages as well. The results in Chapter 6 suggest that unsupervised methods for inducing certain kinds of grammatical relations hold promise, and it will be interesting to see if such methods can accurately predict all types of grammatical relationships, in both English and other languages.

Appendix A

Some Relevant Proofs

A.1 Non-Overlapping Features

As mentioned in Section 2.7.1, we give a proof to show that the probability estimate $p(a|b)$ can be computed in closed form, without need for an iterative algorithm, if the features do not overlap.

Suppose $\mathcal{B}$ is the space of possible contexts, and let $cp_1 \ldots cp_m$ be a set of m predicates that partition $\mathcal{B}$, i.e., $\forall b \in \mathcal{B} \;\; \exists cp \;\; cp(b) = true$ and for any given $b \in \mathcal{B}$, if $cp_1(b) = true$ and $cp_2(b) = true$, then $cp_1 = cp_2$. If $\mathcal{A}$ is the space of possible predictions, assume we have $m|\mathcal{A}|$ features of the form:

$$f_{cp,a'}(a,b) = \begin{cases} 1 & \text{if } a = a' \text{ and } cp(b) = true \\ 0 & \text{otherwise} \end{cases}$$

Here, $f_{cp,a'}$ denotes a feature that tests for the predicate cp together with prediction a'.

Theorem 1 (Non-Overlapping Features). *Let $\mathcal{A}$ denote the possible predictions, and let $\mathcal{B}$ denote space of possible contexts. If the predicates $cp_1 \ldots cp_m$ partition $\mathcal{B}$, and if we are given $m|\mathcal{A}|$ features $f_{cp,a'}$, where $cp \in \{cp_1 \ldots cp_m\}$ and $a' \in \mathcal{A}$, we can use the following closed-form solution for $p(a'|b)$:*

$$p(a'|b) = \frac{E_{\tilde{p}} f_{cp,a'}}{E_{\tilde{p}}[cp(b) = true]}$$

where

$$\begin{aligned} E_{\tilde{p}}[cp(b) = true] &= \sum_b \tilde{p}(b) \sum_a f_{cp,a'}(a,b) \\ E_{\tilde{p}} f_{cp,a'} &= \sum_{a,b} \tilde{p}(a,b) f_{cp,a'}(a,b) \end{aligned}$$

and where cp is the (unique) predicate such that $cp(b) = true$, and where $f_{cp,a'}$ is the (unique) feature that corresponds to the pair a, b.

Proof. We use the fact that the $f_{cp,a'}$ partition $\mathcal{A} \times \mathcal{B}$, and that only one parameter will ever be active on any a, b pair. Rewrite $E_p f_{cp,a'}$:

$$\begin{aligned} E_p f_{cp,a'} &= \sum_{a,b} \tilde{p}(b) p(a|b) f_{cp,a'}(a,b) \\ &= \sum_{a,b: f_{cp,a'}(a,b)=1} \tilde{p}(b) p(a|b) && \text{(A.1)} \\ &= \sum_{a,b: f_{cp,a'}(a,b)=1} \tilde{p}(b) \frac{\alpha_{cp,a'}}{Z_{cp}} && \text{(A.2)} \\ &= \frac{\alpha_{cp,a'}}{Z_{cp}} \sum_{a,b: f_{cp,a'}(a,b)=1} \tilde{p}(b) && \text{(A.3)} \\ &= p(a|b) \sum_{a,b} \tilde{p}(b) f_{cp,a'}(a,b) \end{aligned}$$

Here, $Z_{cp} = \sum_a \alpha_{cp,a}$. Equation A.1 simply re-arranges the summation to only sum over those a, b for which $f_{cp,a'}(a,b) = 1$, and equation A.2 follows from the fact that only one parameter will be used in computing $p(a|b)$, for any a, b, since the $f_{cp,a'}$ partition $\mathcal{A} \times \mathcal{B}$. As a result, $p(a|b)$ can be moved out of the sum, as in A.3. It follows that

$$p(a|b) = \frac{E_{\tilde{p}} f_{cp,a}}{E_{\tilde{p}}[cp(b) = true]}$$

□

A.2 Maximum Likelihood and Maximum Entropy

The purpose of this section is to be a supplement to Chapter 2, in the hopes of making the thesis self-contained. We give proofs that the notions of conditional maximum likelihood and conditional maximum entropy are equivalent under the circumstances discussed in Chapter 2. The proofs given here for conditional models are mostly identical to the proofs for joint models given elsewhere[Della Pietra et al., 1997].

The following few definitions introduce some notation used in the proofs. They assume the existence of a training set

$$\mathcal{T} = (a_1, b_1) \ldots (a_N, b_N)$$

where each element $(a, b) \in \mathcal{T}$ consists of a possible prediction a and a context, or history, b. We use $\tilde{p}(a, b)$ to denote the observed probability of a, b in the set $\mathcal{T}$, and we use $\tilde{p}(b) = \sum_a \tilde{p}(a, b)$ to denote the observed probability of the context b in $\mathcal{T}$. In the proofs that follow, we assume that $\tilde{p}(b) > 0$ for any $b \in B$, but we later discuss the consequences if $\tilde{p}(b) = 0$.

Definition 1 (Relative Entropy over Training Set). *The relative entropy D between two conditional probability distributions p and q is given by:*

$$D(p \parallel q) = \sum_{a,b} \tilde{p}(b) p(a|b) \log \frac{p(a|b)}{q(a|b)}$$

Definition 2 (NonNegativity). *For any two conditional probability distributions p and q,*

$$D(p \parallel q) \geq 0$$

with equality if and only if $p = q$, assuming $\tilde{p}(b) > 0$ for any $b \in \mathcal{B}$.

See [Cover and Thomas, 1991] for a proof.

Definition 3 (Set of consistent probability models). *The set of conditional probability models that are consistent with the k observed feature expectations is denoted by P:*

$$\begin{aligned}
P &= \{p \mid E_p f_j = E_{\tilde{p}} f_j,\ j = \{1 \ldots k\}\} \\
E_{\tilde{p}} f_j &= \sum_{a,b} \tilde{p}(a, b) f_j(a, b) \\
E_p f_j &= \sum_{a,b} \tilde{p}(b) p(a|b) f_j(a, b)
\end{aligned}$$

Definition 4 (Form of log-linear models). *The set of conditional probability models of log-linear form is denoted by* Q*:*

$$\begin{aligned} Q &= \{p \mid p(a|b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)}\} \\ Z(b) &= \sum_a \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \end{aligned}$$

Definition 5 (Entropy over Training Set). *The entropy* $H(p)$ *of a conditional probability distribution* p *over a training set* $\mathcal{T}$ *is defined as:*

$$H(p) = -\sum_{a,b} \tilde{p}(b) p(a|b) \log p(a|b)$$

It is useful to note that $H(p) = -D(p \parallel \pi) + Constant$ where π is the uniform conditional distribution.

Definition 6 (Log-likelihood of Training Set). *The log-likelihood* $L(p)$ *of a conditional probability distribution* p *over a training set* $\mathcal{T}$ *is defined as:*

$$L(p) = \sum_{a,b} \tilde{p}(a,b) \log p(a|b)$$

It is useful to note that $L(p) = -D(\tilde{p} \parallel p) + Constant$.

The following Lemma is called the *Pythagorean property*, since it resembles the Pythagorean theorem if p, q, and p^* represent vertices, and if the relative entropy measure is replaced by the squared distance.

Lemma 1 (Pythagorean Property). *If* $p \in P$*,* $q \in Q$*, and* $p^* \in P \cap Q$*,*

$$D(p \parallel q) = D(p \parallel p^*) + D(p^* \parallel q)$$

Proof. Use the following term for convenience

$$h(p,q) = \sum_{a,b} \tilde{p}(b) p(a|b) \log q(a|b)$$

so that $D(p \parallel q) = h(p,p) - h(p,q)$.

For any $p \in P$, $q \in Q$, rewrite $h(p,q)$:

$$\begin{aligned} h(p,q) &= \sum_{a,b} \tilde{p}(b)p(a|b)[\log \prod_j \alpha_{q,j}^{f_j(a,b)} + \log \frac{1}{Z_q(b)}] \\ &= \sum_j E_p f_j \log \alpha_{q,j} + \sum_b \tilde{p}(b) \log \frac{1}{Z_q(b)} \end{aligned}$$

where the parameters are written as $\alpha_{q,j}$ and $Z_q(b)$ to indicate that they correspond to the probability distribution q.

Also note that for any $p_1, p_2 \in P$, and any $q \in Q$,

$$\begin{aligned} h(p_1,q) &= \sum_j E_{p_1} f_j \log \alpha_{q,j} + \sum_b \tilde{p}(b) \log \frac{1}{Z_q(b)} \\ &= \sum_j E_{p_2} f_j \log \alpha_{q,j} + \sum_b \tilde{p}(b) \log \frac{1}{Z_q(b)} \\ &= h(p_2,q) \end{aligned}$$

Using the above substitution, where $p \in P$, and $q \in Q$, and $p^* \in P \cap Q$, we rewrite $D(p \parallel p^*) + D(p^* \parallel q)$:

$$\begin{aligned} D(p \parallel p^*) + D(p^* \parallel q) &= h(p,p) - h(p,p^*) + h(p^*,p^*) - h(p^*,q) \\ &= h(p,p) - h(p^*,p^*) + h(p^*,p^*) - h(p,q) \\ &= h(p,p) - h(p,q) \\ &= D(p \parallel q) \end{aligned}$$

□

The following two lemmas use Lemma 1 to prove properties about any $p^* \in P \cap Q$. We assume that an exact solution exists, i.e., that $P \cap Q \neq \emptyset$, although we later discuss situations in which this assumption is false.

Lemma 2. *If $p^* \in P \cap Q$, then*

$$p^* = \arg\max_{q \in Q} L(q)$$

Proof. Let $q \in Q$, and show that $L(p^*) - L(q) \geq 0$:

$$\begin{aligned} L(p^*) - L(q) &= D(\tilde{p} \parallel q) - D(\tilde{p} \parallel p^*) \\ &= [D(\tilde{p} \parallel p^*) + D(p^* \parallel q)] - D(\tilde{p} \parallel p^*) \\ &= D(p^* \parallel q) \\ &\geq 0 \end{aligned}$$

$\square$

Lemma 3. *If $p^* \in P \cap Q$, then*

$$p^* = \arg\max_{p \in P} H(p)$$

Proof. Let $p \in P$, and let $\pi \in Q$ be the uniform conditional distribution. Show that $H(p^*) - H(p) \geq 0$:

$$\begin{aligned} H(p^*) - H(p) &= D(p \parallel \pi) - D(p^* \parallel \pi) \\ &= [D(p \parallel p^*) + D(p^* \parallel \pi)] - D(p^* \parallel \pi) \\ &= D(p \parallel p^*) \\ &\geq 0 \end{aligned}$$

$\square$

Theorem 2 shows that the maximum likelihood model of log-linear form is also a maximum entropy model over the set of linear constraints on feature expectations.

Theorem 2 (Maximum Likelihood $\rightarrow$ Maximum Entropy). *If $\hat{p} = \arg\max_{q \in Q} L(q)$, then $\hat{p} = \arg\max_{p \in P} H(p)$*

Proof. If $\hat{p} = \arg\max_{q \in Q} L(q)$ then $L(\hat{p}) - L(q) \geq 0$ for any $q \in Q$. Let $p^* \in P \cap Q$:

$$\begin{aligned} L(\hat{p}) - L(p^*) &= D(\tilde{p} \parallel p^*) - D(\tilde{p} \parallel \hat{p}) \\ &= D(\tilde{p} \parallel p^*) - [D(\tilde{p} \parallel p^*) + D(p^* \parallel \hat{p})] \\ &= -D(p^* \parallel \hat{p}) \\ &\geq 0 \end{aligned}$$

which implies that $D(p^* \parallel \hat{p}) = 0$ and $p^* = \hat{p}$. By Lemma 3, $p^* = \hat{p} = \arg\max_{p \in P} H(p)$.

$\square$

Using similar arguments, we can also show equivalence in the other direction.

Theorem 3 (Maximum Entropy $\rightarrow$ Maximum Likelihood). *If* $\hat{p} = \arg\max_{p \in P} H(p)$ *then* $\hat{p} = \arg\max_{q \in Q} L(q)$

Proof. If $\hat{p} = \arg\max_{p \in P} H(p)$, then $H(\hat{p}) - H(p) \geq 0$ for any $p \in P$. Let $p^* \in P \cap Q$, and let $\pi \in Q$ be the uniform conditional distribution.

$$\begin{aligned}
H(\hat{p}) - H(p^*) &= D(p^* \parallel \pi) - D(\hat{p} \parallel \pi) \\
&= D(p^* \parallel \pi) - [D(\hat{p} \parallel p^*) + D(p^* \parallel \pi)] \\
&= -D(p^* \parallel \hat{p}) \\
&\geq 0
\end{aligned}$$

which implies that $D(p^* \parallel \hat{p}) = 0$ and $p^* = \hat{p}$. By Lemma 2, $p^* = \hat{p} = \arg\max_{q \in Q} L(q)$.

$\square$

Theorems 2 and 3 assume that $\tilde{p}(b) > 0$ for any $b \in \mathcal{B}$, although in practice, this is usually not the case. If $\tilde{p}(b) > 0$ for any $b \in \mathcal{B}$, we can show (using Lemma 1) that the maximum likelihood/maximum entropy solution $p^* \in P \cap Q$ is unique. Otherwise, the solution p^* is not unique and it is be possible to have some $p_1, p_2 \in Q$ that are both maximum likelihood estimates over the contexts of the training set (those b such that $\tilde{p}(b) > 0$), but differ for some context b such that $\tilde{p}(b) = 0$.

Theorems 2 and 3 assume that $P \cap Q$ is non-empty, but the constraints that define P may require the solution to have the value $p(a|b) = 1$ for some pair a, b. No model in Q with finite parameters can achieve a value of 1 (or 0) due to its log-linear form, and hence $P \cap Q$ will be empty, and the exact solution will not exist, leading to the problems discussed in Chapter 9.

[Della Pietra et al., 1997] deal with this problem in theory by using the *closure* of Q, or $\overline{Q}$, and show that it must be the case that $P \cap \overline{Q} \neq \emptyset$. In practice, when the exact solution does not exist, we use the inexact solution returned by the GIS algorithm as a probability model for the data. Chapter 9 discusses why an inexact solution is not likely to cause adverse modeling effects.

Bibliography

[Abney, 1991] Abney, S. (1991). Parsing By Chunks. In Berwick, R., Abney, S., and Tenny, C., editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

[Aho et al., 1988] Aho, A. V., Sethi, R., and Ullman, J. D. (1988). *Compilers : Principles Techniques and Tools*. Addison Wesley.

[Apté et al., 1994] Apté, C., Damerau, F., and Weiss, S. W. (1994). Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems*, 12(3).

[Baayen and Sproat, 1996] Baayen, H. and Sproat, R. (1996). Estimating lexical priors for low-frequency morphologically ambiguous forms. *Computational Linguistics*, 22(2).

[Berger et al., 1996] Berger, A., Della Pietra, S. A., and Della Pietra, V. J. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

[Black et al., 1991] Black, E. et al. (1991). A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*, pages 306–311.

[Black et al., 1993] Black, E., Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., and Roukos, S. (1993). Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the 31st Annual Meeting of the ACL*, Columbus, Ohio.

[Breiman et al., 1982] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1982). *Classification and Regression Trees.* Wadsworth, Belmont.

[Brill, 1993a] Brill, E. (1993a). *A Corpus-Based Approach to Language Learning.* PhD thesis, University of Pennsylvania.

[Brill, 1993b] Brill, E. (1993b). Transformation-Based Error-Driven Parsing. In *Proceedings of the 31st Annual Meeting of the ACL*, Columbus, Ohio.

[Brill, 1994] Brill, E. (1994). Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 722–727.

[Brill and Resnik, 1994] Brill, E. and Resnik, P. (1994). A Rule Based Approach to Prepositional Phrase Attachment Disambiguation. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING).*

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

[Brown et al., 1992] Brown, P. F., Della Pietra, V., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

[Bruce and Weibe, 1994] Bruce, R. and Weibe, J. (1994). Word-Sense Disambiguation Using Decomposable Models. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 139–146.

[Charniak, 1997] Charniak, E. (1997). Statistical Parsing with a Context-free Grammar and Word Statistics. In *Fourteenth National Conference on Artificial Intelligence*, Providence, Rhode Island.

[Church, 1988] Church, K. (1988). A stochastic parts program and noun phrase chunker for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing.*

[Collins, 1997] Collins, M. (1997). Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL, and 8th Conference of the EACL*, Madrid, Spain. ACL.

[Collins and Brooks, 1995] Collins, M. and Brooks, J. (1995). Prepositional Phrase Attachment through a Backed-off Model. In Yarowsky, D. and Church, K., editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38, Cambridge, Massachusetts.

[Collins, 1996] Collins, M. J. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the ACL.*

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.* Wiley, New York.

[Csiszar, 1975] Csiszar, I. (1975). I-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1):146–158.

[Csiszar, 1989] Csiszar, I. (1989). A Geometric Interpretation of Darroch and Ratcliff's Generalized Iterative Scaling. *The Annals of Statistics*, 17(3):1409–1413.

[Darroch and Ratcliff, 1972] Darroch, J. N. and Ratcliff, D. (1972). Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

[de Lima, 1997] de Lima, E. F. (1997). Assigning Grammatical Relations with a Back-off Model. In Cardie, C. and Weischedel, R., editors, *Second Conference on Empirical Methods in Natural Language Processing*, Providence, R.I.

[Della Pietra et al., 1997] Della Pietra, S., Della Pietra, V., and Lafferty, J. (1997). Inducing Features of Random Fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19(4).

[Eisner, 1997] Eisner, J. M. (1997). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Copenhagen, Denmark.

[Francis and Kucera, 1982] Francis, W. N. and Kucera, H. (1982). *Frequency analysis of English usage: lexicon and grammar.* Houghton Mifflin, Boston.

[Franz, 1997] Franz, A. (1997). Independence Assumptions Considered Harmful. In *Proceedings of the 35th Annual Meeting of the ACL, and 8th Conference of the EACL*, Madrid, Spain. ACL.

[Gale et al., 1992] Gale, W., Church, K., and Yarowsky, D. (1992). A method for disambiguating word senses in large corpus. *Computers and the Humanities*, 26:415–439.

[Good, 1963] Good, I. J. (1963). Maximum Entropy for Hypothesis Formulation, Especially for Multidimensional Contingency Tables. *The Annals of Mathematical Statistics*, 34:911–934.

[Goodman, 1997] Goodman, J. (1997). Probabilistic feature grammars. In *Proceedings of the International Workshop on Parsing Technologies.*

[Hermjakob and Mooney, 1997] Hermjakob, U. and Mooney, R. J. (1997). Learning Parse and Translation Decision From Examples With Rich Context. In *Proceedings of the 35th Annual Meeting of the ACL, and 8th Conference of the EACL*, Madrid, Spain. ACL.

[Hindle and Rooth, 1993] Hindle, D. and Rooth, M. (1993). Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19(1):103–120.

[Hosmer and Lemeshow, 1989] Hosmer, Jr., D. W. and Lemeshow, S. (1989). *Applied Logistic Regression.* Wiley, New York.

[Jaynes, 1957] Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *Physical Review*, 106:620–630.

[Jelinek, 1990] Jelinek, F. (1990). Self-Organized Language Modeling for Speech Recognition. In Waibel, A. and Lee, K.-F., editors, *Readings in Speech Recognition.* Morgan Kaufmann.

[Jelinek et al., 1994] Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., Ratnaparkhi, A., and Roukos, S. (1994). Decision Tree Parsing using a Hidden Derivational Model. In

Proceedings of the Human Language Technology Workshop, pages 272–277, Plainsboro, N.J. ARPA.

[Johansson, 1986] Johansson, S. (1986). *The Tagged LOB Corpus: User's Manual.* Norwegian Computing Centre for the Humanities, Bergen, Norway.

[Karp et al., 1992] Karp, D., Schabes, Y., Zaidel, M., and Egedi, D. (1992). A freely available wide coverage morphological analyzer for english. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING)*, Nantes, France.

[Katz, 1987] Katz, S. M. (1987). Estimation of probabilities from sparse data for language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3).

[Kayaalp et al., 1997] Kayaalp, M., Pedersen, T., and Bruce, R. (1997). A statistical decision making method: A case study on prepositional phrase attachment. In *Computational Natural Language Learning Workshop*, Madrid, Spain.

[Larsen and Marx, 1986] Larsen, R. J. and Marx, M. L. (1986). *An Introduction to Mathematical Statistics and its Applications.* Prentice-Hall.

[Lau, 1994] Lau, R. (1994). Adaptive statistical language modelling. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

[Lau et al., 1993] Lau, R., Rosenfeld, R., and Roukos, S. (1993). Adaptive Language Modeling Using The Maximum Entropy Principle. In *Proceedings of the Human Language Technology Workshop*, pages 108–113. ARPA.

[Lewis, 1992] Lewis, D. D. (1992). *Representation and Learning in Information Retrieval.* PhD thesis, University of Massachusetts, Amherst, Mass.

[Lewis and Ringuette, 1994] Lewis, D. D. and Ringuette, M. (1994). A Comparison of Two Learning Algorithms for Text Categorization. In *Third Annual Symposium of Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, Nevada.

[Liberman and Church, 1992] Liberman, M. Y. and Church, K. W. (1992). Text analysis and word pronunciation in text-to-speech synthesis. In Furui, S. and Sondi, M. M., editors, *Advances in Speech Signal Processing*. Marcel Dekker, Inc., New York.

[Magerman, 1995] Magerman, D. M. (1995). Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the ACL*.

[Marcus, 1980] Marcus, M. P. (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, Mass.

[Marcus et al., 1994] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1994). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[McNemar, 1969] McNemar, Q. (1969). *Psychological Statistics*. John Wiley and Sons, Inc.

[Merialdo, 1994] Merialdo, B. (1994). Tagging English Text with a Probabilistic Model. *Computational Linguistics*, 20(2):155–172.

[Merlo et al., 1997] Merlo, P., Crocker, M. W., and Berthouzoz, C. (1997). Attaching Multiple Prepositional Phrases: Generalized Backed-off Estimation. In Cardie, C. and Weischedel, R., editors, *Second Conference on Empirical Methods in Natural Language Processing*, pages 149–155, Providence, R.I.

[Nunberg, 1990] Nunberg, G. (1990). The Linguistics of Punctuation. Technical Report C.S.L.I. Lecture Notes, Number 18, Center for the Study of Language and Information.

[Palmer and Hearst, 1997] Palmer, D. D. and Hearst, M. A. (1997). Adaptive Multilingual Sentence Boundary Disambiguation. *Computational Linguistics*, 23(2):241–267.

[Pedersen et al., 1997] Pedersen, T., Bruce, R., and Wiebe, J. (1997). Sequential Model Selection for Word Sense Disambiguation. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 388–395, Washington D.C.

[Quinlan, 1986] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Quinlan, 1992] Quinlan, J. (1992). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

[Ramshaw and Marcus, 1995] Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning. In Yarowsky, D. and Church, K., editors, *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, Massachusetts.

[Ratnaparkhi, 1996] Ratnaparkhi, A. (1996). A Maximum Entropy Part of Speech Tagger. In Brill, E. and Church, K., editors, *Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania.

[Ratnaparkhi et al., 1994a] Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994a). A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the Human Language Technology Workshop*, pages 250–255, Plainsboro, N.J. ARPA.

[Ratnaparkhi et al., 1994b] Ratnaparkhi, A., Roukos, S., and Ward, R. T. (1994b). A Maximum Entropy Model For Parsing. In *Proceedings of the International Conference on Spoken Language Processing*, pages 803–806, Yokohama, Japan.

[Reynar and Ratnaparkhi, 1997] Reynar, J. C. and Ratnaparkhi, A. (1997). A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington D.C.

[Riley, 1989] Riley, M. D. (1989). Some applications of tree-based modelling to speech and language. In *Proceedings of the DARPA Speech and Language Technology Workshop*, pages 339–352, Cape Cod, Massachusetts. DARPA.

[Rosenfeld, 1996] Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech, and Language*, 10.

[Sánchez-León, 1994] Sánchez-León, F. (1994). Spanish tagset for the CRATER project. Technical report, Laboratorio de Lingüística Informática, Universidad Autónoma de Madrid. Also available on cmp-lg archive as `http://xxx.lanl.gov/cmp-lg/9406023`.

[Sekine, 1997] Sekine, S. (1997). The Domain Dependence of Parsing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 96–102, Washington D.C.

[Stetina and Nagao, 1997] Stetina, J. and Nagao, M. (1997). Corpus Based PP Attachment Ambiguity Resolution with a Semantic Dictionary. In Zhou, J. and Church, K., editors, *Proceedings of the Fifth Workshop on Very Large Corpora*, pages 66–80, Beijing and Hong Kong.

[van Halteren et al., 1998] van Halteren, H., Zavrel, J., and Daelemans, W. (1998). Improving data driven wordclass tagging by system combination. In *Proceedings of the Seventeenth International Conference on Computational Linguistics (COLING-ACL)*, University of Montreal.

[Weischedel et al., 1993] Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., and Palmucci, J. (1993). Coping With Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359–382.

[White, 1995] White, M. (1995). Presenting punctuation. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 107–125, Leiden, The Netherlands.

[Yarowsky, 1996] Yarowsky, D. (1996). *Three Machine Learning Algorithms for Lexical Ambiguity Resolution*. PhD thesis, University of Pennsylvania.

[Zavrel and Daelemans, 1997] Zavrel, J. and Daelemans, W. (1997). Memory-Based Learning: Using Similarity for Smoothing. In *Proceedings of the 35th Annual Meeting of the ACL, and 8th Conference of the EACL*, Madrid, Spain. ACL.