Maximum Entropy Models and Prepositional Phrase Ambiguity

Mark McLauchlan

Master of Science

School of Artificial Intelligence
Division of Informatics
University of Edinburgh

2001

Abstract

Prepositional phrases are a common source of ambiguity in natural language and many approaches have been devised to resolve this ambiguity automatically. In particular, several different machine learning approaches have now reached accuracy rates of around 84.5% on the benchmark dataset. Maximum entropy (maxent) models, despite their successful application in many other areas of natural language processing, have only reached 83.7% accuracy on this task. This dissertation shows that maxent models can achieve accuracy rates of 84.7% using standard features, rising to 85.3% when using additional features based on Latent Semantic Analysis (LSA) information. Three feature selection techniques are compared in this domain: frequency cut-off, information gain (mutual information) and a new method that uses the variation of feature weights across several training sets. A simple frequency cut-off is found to be the most robust method across of a variety of models. We also consider ensembles of maxent classifiers created using bagging and noisy bagging, a more effective variant where additional random noise is added to each training set. The latter approach creates a more diverse set of classifiers and a maxent ensemble using noisy bagging achieves 85.53% accuracy.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Mark McLauchlan*)

Table of Contents

Chapter 1

Introduction

1.1 Introduction

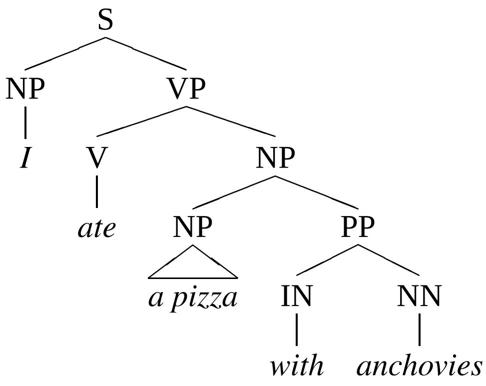
This dissertation examines the application of maximum entropy models for resolving prepositional phrase attachment ambiguity. Prepositional phrases (PP's) are very common constructs in English:

(1.1) I ate a pizza with anchovies

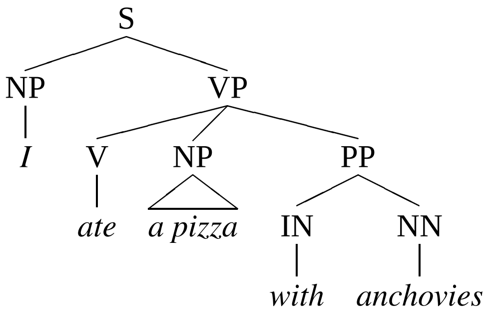
(1.2) I ate a pizza with friends

On the surface these sentences are quite similar but they have important semantic (and flavour) differences. The root of the ambiguity is the prepositional phrase headed by the preposition "with" — in each case we have to decide whether it belongs with the verb ("ate") or the noun phrases ("a pizza"). Each interpretation corresponds to a different grouping of the constituent phrases, a different application of English syntax rules and ultimately a different meaning. The two possible syntactic structures for sentence (1.1) are formalised using the following two *parse trees*:

(1.3)



(1.4)



The various symbols (NP, PP, etc) are a shorthand for different phrases (noun phrase, prepositional phrase etc). In this case, the first tree (1.3) is the correct parse: the prepositional noun phrase attaches to the object noun phrase, indicating that the anchovies belong with the pizza rather than the eating.

Humans have no trouble reaching the correct conclusion, but the goal of natural language processing is to create computer systems capable of the same feat. Creating systems to automatically determine the correct syntactic interpretation is an important step towards that goal. Unfortunately, as the above example shows, there are often multiple possible syntactic structures even for a simple sentence. The presence of such *structural ambiguity* means that the automatic parser must have some way of choosing the most likely interpretation from a pool of possible candidates. This dissertation presents techniques for resolving one particular kind of structural ambiguity, namely ambiguity arising from the presence of prepositional phrases.

1.2 Defining the problem

This section outlines some general properties of this automatic disambiguation system:

- **Inputs** What informatin does it require?
- **Outputs** What do we want in return?
- **Upper bounds** How successful can we expect the system to be?

There is a lot of information that the system might conceivably require to make its decision. Some authors [Altman and Steedman, 1988] have suggested that nothing less than a model of the world and the discourse structure is required. Happily it turns out that the system can achieve reasonable accuracy without such extensive information, but the question about exactly how much context should be included remains. The sentence (1.1) can be resolved out of context, but this is not the case for (1.7).

The system should output the correct attachment site of the PP. Unfortunately, PP's are surprisingly adaptable and there often a potentially large number of attachment sites. We have already seen examples of noun (1.3) and verb (1.4) attachment, but the possibilities to not stop there. Consider the following examples from [Hindle and Rooth, 1993] and [Franz, 1996] respectively:

(1.5) NBC was so afraid of hostile advocacy groups and unnerving advertisers that it shot its dramatization of the landmark court case that legalised abortion [under two phony script titles].

(1.6) Temperatures above 80 degrees can be harmful [to the unit].

In the first example, the marked prepositional phrase attaches not to the previous verb or noun (*legalised* and *abortion*) but to a higher verb (*shot*). In the second, the PP attaches to an adverb (*harmful*).

To simplify matters, most authors have only considered cases where the PP modifies the immediately preceding noun or verb; these are also by far the most common cases [Franz, 1996]. Several authors [Gibson and Pearlmutter, 1994, Chen and Chen, 1996] have suggested that these cases are too easy and such a system would be too limited to be useful. A few authors [Franz, 1996] and [Chen and Chen, 1996] have taken up this challenge and produced systems that could handle more complex sentences with additional attachment sites. This dissertation follows tradition and only considers noun and verb attachments, but our system can be easily adapted to the more general case.

Our assertion that humans resolve this ambiguity effortlessly is not completely true, since there are many examples where the correct attachment site is not so obvious:

(1.7) I want that car in the picture

(1.8) I saw the man with the telescope

[Hindle and Rooth, 1993] categorise a number of highly ambiguous PP types: (the following list is adapted from [Franz, 1996]):

- **Idioms** Certain idiomatic phrases contain nouns that cannot properly considered to be valid attachment sites: "... visited from time to time", "... carried all but one".
- **Light verb constructions** A similar situation occurs with verbs that have low semantic content: "... made cuts in Social Security".
- **Locative ambiguity** If something happens in a place, then both the action and the thing are in that place: "... searched the galleries in London", or "... was a Friday in June".
- **Benefactive ambiguity** If something is intended for someone, then the thing is also for them: "... prepared the report for the supervisor".

Luckily it appears that such highly ambiguous prepositional phrases are rare (see Section 8.3), but they will impose an upper limit on the accuracy of the final system.

1.3 PP ambiguity resolution as a machine learning task

The methods used in this dissertation for resolving ambiguity originate from the machine learning community, where this task is best described as a *classification* task.

Classification tasks have the following general properties:

1. **Inputs** We have a set of examples X, and each individual example x is defined by a set of feature-value pairs, $x_1 = i, x_2 = j, \ldots$. In this case, each example is an instance of a prepositional phrase.

2. **Outputs** There is a known, discrete set of classes A. For this task, each example x belongs to exactly one class a, where a represents a particular attachment site. We only consider two possible classes: $A = VA$ (verb attachment) or $A = NA$ (noun attachment).

3. **The system** We want to build a *classifier* that takes an example x as input and produces a predicted class a as output.

This dissertation uses the *maximum entropy* (maxent) framework to build classifiers. Maximum entropy modelling is a statistically motivated method for modelling the probability distribution $p(A|X)$, or the probability that an example x belongs to a class a. A classifier built using this model assigns the example to the most probable class. Maxent has been successfully applied to several other natural language tasks, including Part-Of-Speech tagging [Ratnaparkhi, 1996], machine translation [Berger et al., 1996], and language modelling [Lau, 1994], yet the results of using maxent for the PP attachment task have been somewhat disappointing.

The first part of this dissertation aims to show that maxent classifiers can achieve comparable accuracy in this domain. Chapter 3 presents an overview of the maxent framework and illustrates its application to the PP attachment problem. We describe a maxent classifier with 84.79% accuracy, higher than the previous best published maxent result of 83.7% and slightly higher than other comparable approaches achieving around 84.5% accuracy. In Chapter 4 we compare two maxent-based classifiers: one using a single underlying model and another using a series of maxent models combined using linear interpolation[1].

[1] Linear interpolation is used a method of smoothing probability estimates by appealing to successively simpler models, and is used by one of the most successful approaches to this task

In Chapters 5 and 6 we consider what features we should use to make the attachment decision. Most approaches use purely lexical information, so each feature is a syntactic position such as *preposition* and the set of values consists of all possible words or phrases that can be used in that position. In Chapter 5 we consider several new feature types and present a new maxent classifier that achieves 85.34% accuracy. These new features use *unsupervised* data, meaning that the classifier needs no human-created data other than that in the training set. Chapter 6 then considers some techniques for removing redundant or unnecessary features from the model to combat the overfitting problem and improve its accuracy rate.

Chapter 7 considers the proposition that ensembles or collections of classifiers can outperform an equivalent single classifier. Two methods for creating ensembles of maxent classifiers are presented: bagging and a more effective variant, noisy bagging. An ensemble of classifiers created using noisy bagging achieves a mean accuracy rate of 85.53% accuracy. Chapter 8 analyses the errors made by a standard maxent classifier and summarises the key results of this dissertation.

Chapter 2

Previous Work

2.1 There is more than one way to attach a preposition

This chapter reviews some key papers in the PP attachment field. These papers differ chiefly in what information the authors believe to be most important for disambiguation. There are four main categories:

- **syntactic information** the grammatical structure associated with each parse.
- **semantic information** the underlying meaning attached to words, as specified by humans.
- **pragmatic information** the relationship between words and real-world entities or other elements in the discourse.
- **lexical information** actual words.

We briefly review previous work in the first three categories above, and then delve more deeply into lexical strategies — the category in which this project belongs.

2.2 Using syntax

Early work on the problem used purely syntactic knowledge to resolve any ambiguity, meaning the atachment decision is made just by comparing the structure of the candidate parse trees. The attachment that produces the most appealing parse tree is selected. These approaches do not yield particularly high accuracy rates (see below), although in fairness they were designed as models of the human parsing mechanism rather than robust computational models.

The two most frequently cited strategies are *Right Association* [Kimball, 1973] and *Minimal Attachment* [Frazier, 1979]. The former expresses a preference for right-branching trees, so terminal symbols should aim to attach to the lowest rightmost non-terminal node. The second is based on a preference for small parse trees. However, in the case of simple PP attachment, these have contradictory suggestions: right association always attaches the PP to the noun while minimal attachment usually specifies the verb. A study by [Whittemore et al., 1990] showed that neither scheme is particularly satisfactory: Right Association was the best with an accuracy of 55% on a test set of 724 phrases; a strict application of Minimal Attachment had an accuracy rate of only 36% on a test set of 488 phrases.

2.3 Using semantics

The PP attachment task has also been investigated by researchers interested in automatic word-sense disambiguation and semantic acquisition. The goal of this research is to create a system for identifying the correct meaning of a word and grouping semantically similar words together. We can use this information to decide on the most probable parse. For example, in our canonical prepositional phrase:

(2.1) eat pizza with anchovies

each word is first replaced with the most appropriate semantic sense. A system with semantic knowledge of *pizza* should be able to correctly disambiguate this sentence, since an interpretation relating the two foodstuffs is more probable. There is considerable overlap between semantic an lexical approaches, and some lexically-based approaches reviewed in that also use semantic information.

2.4 Using pragmatics and discourse information

[Altman and Steedman, 1988] suggested that PP ambiguity can only be solved if we use additional pragmatic and discourse information. They propose a set of principles, for example:

- **A Priori Plausibility** If an interpretation is more plausible in terms either of general knowledge about the world, or of specific knowledge about the universe of discourse, then, everything else being equal, it is favoured over an interpretation that is not as plausible.

This kind of real-world information may indeed be required to achieve 100% accuracy. Consider the following sentences:

(2.2) Buy a car with a steering wheel

(2.3) I want that car in the picture

A classifier with pragmatic information could disambiguate sentence (2.2) by leveraging its knowledge about cars and deducing that steering wheels are usually parts of cars and not a means of exchange. The second example (2.3) is hopelessly ambiguous by itself but it should be resolvable with knowledge about the surrounding discourse. Unfortunately, while the Altman & Steedman rules may point out the necessary ideas for disambiguation (although [Gibson, 1991] suggests it is still inadequate), research into world models and discourse analysis is arguably decades away from producing computationally feasible implementations.

2.5 Using lexical information

Luckily, it turns out that we can ignore semantics and pragmatics completely and still obtain good accuracy by using words alone as features. [Ford et al., 1982] proposed the idea of *lexical preferences* — that we should prefer some parses over others on the basis of the words involved. These preferences are usually derived empirically from some large corpus. For example, lexical information should be sufficient to resolve (2.2) since steering wheels are more commonly associated with cars rather than as a means of exchange. The Whittemore study of various attachment techniques [Whittemore et al., 1990] was one of the first to apply lexical preferences to PP attachment, and their success has motivated all subsequent corpus-based work. But while Whittemore crafted a set of lexical preferences by hand, later algorithms compute them automatically using a large corpus of training material. These approaches can be categorised along three dimensions:

- **Data** There is usually a division between approaches where the data has already been tagged with the correct attachment decision (supervised learning) and where it has not (unsupervised learning). Supervision is a relative term, and most so-called unsupervised approaches still have some degree of outside help. The interesting aspect is the degree of supervision.

- **Features** How much context does each approach use for disambiguation? If the project uses word classes (groups of words having similar semantic or other common properties), what features are the classes based on?

- **Learning algorithm** A learning algorithm is a method for constructing a classifier. Classification and machine learning have been extensively studied and there are many existing algorithms that can be pressed into service for this task.

Before delving into the various approaches, it is worthwhile mentioning the supervised dataset created by [Ratnaparkhi et al., 1994] which has emerged as a benchmark in the field. It consists of 27,937 potentially ambiguous sentence fragments from the hand-parsed Wall Street Journal corpus where a prepositional noun phrase occurred after a verb phrase and a noun phrase. Each sentence fragment is broken up into four constituent phrases (V, N_1, P, N_2) and these are reduced to their *headwords* only. The headword is the syntactically most important word in the phrase, for example *pizza* is the headword of the phrase *the jumbo pizza*. The example (2.4) below (adapted from the WSJ) would be transformed into the 4-tuple (expects, decline, in, rates). Example (2.5) (also from the WSJ) would give two 4-tuples: (dumped, sacks, of, material) and (dumped, sacks, into, bin). The dataset is divided into 20,801 items for training, 4039 items for validation and testing during development, and 3097 items for final testing.

(2.4) The Treasury expects a decline in interest rates

(2.5) Workers dumped sacks of the material into a bin

The Wall Street Journal covers a fairly wide domain and this dataset is extremely sparse. Only 5% of the 4-tuples in the testing set occur in the same form in the training set. The dataset is also known to be fairly noisy: human experts only achieve 88.2% accuracy on the task [Ratnaparkhi et al., 1994]. This is partly due to inherent ambiguity (see Section 1.2) but also because the restricted feature set makes some examples ambiguous or simply nonsensical — human experts achieve 93.2% accuracy when given access to the whole sentence. Figure 2.1 lists some examples from this corpus, roughly in order of increasing ambiguity, along with their nominal classification.

The following sections outline several key papers in the field, grouped according to learning algorithm. A summary of results can be found in Table 2.1. We distinguish between approaches that only use the labelled training data, and those that have additional supervision of some form. The most common form of extra supervision is WordNet [Miller, 1990], a hand-crafted dictionary for English (see Section 5.2.3).

2.5.1 Maximum Likelihood-based classifiers

Maximum likelihood estimation is a method of estimating the parameters of a probabilistic model from empirical data. From a probabilistic point of view, we are interested in the conditional distribution $p(A|X)$ where A and X are random variables representing the attachment site and the context respectively. Digressing briefly, note that all

Algorithm	Accuracy	(RRR94) Dataset?	Extra Supervision?
Always noun	55.0	Yes	
Most likely for each P	72.19	Yes	
Average human, headwords	88.2	Yes	
Average human, whole sentence	93.2	Yes	
(HR93) ML-based	79.7		
(RRR94) Maxent, words	77.7	Yes	
(RRR94) Maxent, words & classes	81.6	Yes	
(RRR94) Decision trees	77.7	Yes	
(BR94) Trans.-Based Learning	81.8		WordNet
(CB95) ML-based	84.5	Yes	
(Fra96) Loglinear model	82		
(SN97) decision trees	88.1	Yes	WordNet
(ZDV97) Memory-Based Learning	84.4	Yes	LexSpace
(Rat98) Maxent, unsupervised	81.9		
(Rat98) Maxent, supervised	83.7	Yes	
(ASL99) Neural Nets	86.0	Yes	WordNet
(ASS99) Boosting	84.4	Yes	
(PL00) Semi-probabilistic	84.31	Yes	
Maxent, standard features	84.71	Yes	
Maxent, extra features	85.3	Yes	LSA
Maxent, ensemble	85.5	Yes	LSA

Table 2.1: PPA league table

```
retain post as publisher N
restructure itself like corporation V
leaves decisions to executives V
report loss for quarter N
received report from Agency N
struck deal with leaders N
yield % in 2004 N
do something about this V
takes years to years N
was problem for me V
sing birthday to you N
had damage beyond trouble N
given smoking than blacks V
```

Figure 2.1: Example 4-tuples from the [Ratnaparkhi et al., 1994] dataset

random variables are written in upper case, but specific instantiations of those variables are in lower case. For example, V is the class of all verbs, but v is a specific verb. The set of attributes of X usually includes some or all of the PP 4-tuple (V,N_1,P,N_2) and in some cases additional information deemed to be relevant. The classifier chooses the attachment a such that $a = \operatorname{argmax}_a p(a|x)$.

Given a set of training examples (the empirical data), the maximum likelihood estimate of $p(a|x)$ is:

$$\tilde{p}(a|x) = \frac{f(a,x)}{f(x)} \tag{2.6}$$

where f is the frequency function: $f(x)$ measures the number of times that the context x occurred in the training corpus. $\tilde{p}(A|X)$ is the ML estimate of the true distribution $p(A|X)$.

Vanilla maximum likelihood estimation produces a classifier that is most consistent with our training data. There are two problems with this:

- **Unknown words** We may not have enough training data to see all possible contexts. This *sparse data* problem is endemic in natural language applications. If the classifier encounters a completely new context x that did not occur in the training data then $\tilde{p}(a|x) = 0$. For example, the PP quadruple (eat, pizza, with, anchovies) is perfectly plausible, but if it wasn't in the training set then ML estimation considers it to be impossible. This means the estimates are usually *smoothed* in some way to assign non-zero estimates for plausible unseen contexts. A simple smoothing scheme is to add 1 to the counts of all contexts, including a special "unknown" context, thereby setting aside a small amount of probability for all unseen words.

- **Noise** Noise in the training set means spurious regularities that the classifier should ignore. Noise arises naturally in this domain through ambiguous examples or annotation error. For example, if a sentence containing (eat, pizza, with, anchovies) did occur once in the dataset but as part of some odd context where the fish was used as an eating instrument, then the ML model will have $p(A = VA|eat, pizza, with, anchovies) = 1$. Once again, probability estimates can be smoothed in an attempt to iron out noise.

Maximum likelihood-based methods can therefore be analysed in terms of the context used to create the classifier and the smoothing methods used to overcome the above problems. Note that a smoothed estimate is no longer the true maximum likelihood estimate.

2.5.1.1 Hindle and Rooth 1993

The pioneering work in this field is generally considered to be [Hindle and Rooth, 1993]. This approach essentially considers whether the preposition is more likely to co-occur with the noun or the verb:

$$\tilde{p}(VA|v) = \frac{f(p,v)}{f(v)} \tag{2.7}$$

$$\tilde{p}(NA|n) = \frac{f(p,n)}{f(n)} \tag{2.8}$$

The counts were also smoothed using the general attachment preference for each preposition:

$$\tilde{p}(VA|v) = \frac{f(p,v) + \frac{f(p,V)}{f(V)}}{f(V)+T} \tag{2.9}$$

$$\tilde{p}(NA|n_1) = \frac{f(p,n_1) + \frac{f(p,N)}{f(N)}}{f(N)+T} \tag{2.10}$$

where $f(p,V) = \sum_v f(n,v)$, $f(V) = \sum_v f(v)$ and equivalently for noun frequencies, and T is the total number of observed tuples.

This formulation has the advantage of only considering (V, P) and (N_1, P) tuples, reducing the sparse data problem. The downside is that using fewer context words may harm accuracy. One obvious problem is that the prepositional noun (N_2) is ignored completely, so this approach could not correctly disambiguate both the pizza examples given in the Introduction. Later approaches rectify this problem, and in this dissertation (Chapter 5) we also show that attributes expressing the likelihood of (V,P) and (N_1,P) tuples are a useful addition to the model.

This approach predates the Ratnaparkhi dataset, and the classifier is trained using partially supervised data extracted with an automatic parser. A set of heuristics identifies an initial set of unambiguous cases and these are used to produce initial attachment estimates. The algorithm iteratively improves its estimates by adding to the training data any new examples that are classified with high confidence, and re-estimating the probability distributions. The algorithm scored around 80% on a test set of 880 hand-labelled examples. Human experts scored 85-88% on the same test set.

2.5.1.2 Back-off Models: Collins and Brooks 1995

The approach described by Collins & Brooks in this paper continues in the same vein as Hindle and Rooth's, but attacks the sparse data problem in a different and more effective way. This algorithm also uses the explicitly supervised Ratnaparkhi dataset. Hindle & Rooth handled the sparse data problem by using very little context in their classifier. Collins & Brooks use the full PP 4-tuple, but only if there is enough data to support a maximum likelihood estimate — that is, if $f(V,N_1,P,N_2) > c$. Otherwise, a simpler distribution such as $p(A|V,N_1,P)$ is consulted instead. This idea is inspired by Katz's back-off algorithm [Katz, 1987], a commonly used smoothing technique in language modelling.

This backing-off process is conceptually very simple. The algorithm proceeds as follows:

1. If $f(v,n_1,p,n_2) > c_4$ then

   $$\tilde{p}(1|v,n_1,p,n_2) = \frac{f(1,v,n_1,p,n_2)}{f(v,n_1,p,n_2)}$$

   In other words, if the tuple (v, n_1, p, n_2) was seen more than c_4 times in the training data, then the classifier used to predict the attachment uses the standard maximum likelihood estimate.

2. Else if $f(v,n_1,p) + f(v,p,n_2) + f(n_1,p,n_2) > c_3$ then

   $$\tilde{p}(1|v,n_1,p,n_2) = \frac{f(1,v,n_1,p) + f(1,v,p,n_2) + f(1,n_1,p,n_2)}{f(v,n_1,p) + f(v,p,n_2) + f(n_1,p,n_2)}$$

   If the 4-tuple was not seen in the training data then the algorithm backs off to a 3-tuple instead. Instead of choosing one particular 3-tuple to use, the classifier essentially uses an average of three 3-tuples. This is no longer a true maximum likelihood estimate; it can be viewed as a smoothed estimate taken from three different models. The 3-tuple without a preposition is excluded from the model since it has much less predictive power and including it harms the accuracy of the model. All preposition-less 2-tuples are also omitted:

Stage	Total Number	Percent Correct	Median Frequency
4-Tuples	242	92.6	2
3-Tuples	977	87.8	2
2-Tuples	1739	82.4	3
1-Tuples	136	72.8	3500
Default	3	100	
Total	3097	84.5	

Table 2.2: A stage-wise analysis of the Collins & Brooks algorithm

3. Else if $f(v,p)+f(n_1,p)+f(n_2,p) > c_2$ then

$$\tilde{p}(1|v,n_1,p,n_2) = \frac{f(1,v,n_1,p)+f(1,v,p,n_2)+f(1,n_1,p,n_2)}{f(v,n_1,p)+f(v,p,n_2)+f(n_1,p,n_2)}$$

4. Else if $f(p) > c_1$ then

$$\tilde{p}(1|v,n_1,p,n_2) = \frac{f(1,v,n_1,p)+f(1,v,p,n_2)+f(1,n_1,p,n_2)}{f(v,n_1,p)+f(v,p,n_2)+f(n_1,p,n_2)}$$

5. Else attach to the noun (the default)

The best results were obtained when all $c_i = 0$ for all i, implying that in this domain any estimate using more context is better than backing off to a simpler estimate. The accuracy of the classifier at each back-off stage is shown in Table 2.2. The median frequencies are included to give an idea of how much data is typically used for each estimate. These figures are taken from a re-implementation of this approach by the author.

These results are after the dataset was processed using some basic morphological filter (see Section 4.3.1) in an effort to further reduce the sparse data problem. When trained on unprocessed data the accuracy falls to 84.1%. A backed-off classifier using maxent models achieves 84.37% accuracy, rising to 84.63% if we use a more general backing-off procedure (see Chapter 4).

2.5.1.3 Unsupervised ML-based: Pantel and Lin 2000

Instead of using the rather small Ratnaparkhi dataset, this approach uses an automatic parser to extract a very large training set of PP examples. As with Hindle & Rooth, there is an initial training set of 4.4 million unambiguous prepositional phrases, where a PP with only a verb or a noun preceding it, and an additional set of 8.9 million ambiguous PP's. These are resolved iteratively using a heuristic algorithm. Given a new (v,n_1,p,n_2) quadruple, the classifier heuristically scores the triples (v,p,n_2) and (n_1,p,n_2) and classifies the example as a case of verb attachment if the (v,p,n_2) score is higher. Six tuples are considered in the decision:

- (P)
- (V,P), (N_1,P), (P,N_2)
- (V,P,N_2), (N_1,P,N_2)

The score of a particular verb-centric (v,p,n_2) or noun-centric (n_1,p,n_2) triple is initially based on its frequency in the training set, adjusted by the number of the nearby attachment locations. These adjusted frequencies are then iteratively updated using the following ratio (this example is for verb triples):

$$f(v,p,n_2) = \frac{VScore(v,p,n_2)}{\sum_i VScore(v_i,p_i,n_2i)} \tag{2.11}$$

An equivalent ratio using *NScore* is used to update the (n_1,p,n_2) frequencies. *NScore* and *VScore* are the log probabilities $p(VA|V,P,N_2)$ and $p(NA|N_1,P,N_2)$, factored in the following way:

$$p(VA|V,P,N_2) \approx p(P)p(V,P,N_2)p(V,P|V)p(P,N_2|N_2) \tag{2.12}$$

$$p(NA|N_1,P,N_2) \approx p(P)p(N_1,P,N_2)p(N_1,P|N)p(P,N_2|N_2) \tag{2.13}$$

Each term is estimated in the usual maximum likelihood way, that is (for example):

$$\tilde{p}(V,P|V) = \frac{f'(V,P)}{f'(V)} \tag{2.14}$$

These are smoothed versions of the maximum likelihood estimate since the frequency function f' uses adjusted frequencies described above, it is not clear whether these are still true probability estimates although they are referred to as such in the paper. The final score of a (v,p,n_2) and (n_1,p,n_2) tuples are based on a combination of the initial adjusted frequencies and frequencies updated using the above ratio. This approach achieves 84.11% accuracy, rising to 84.31% if the *NScore* and *VScore* functions are extended to compute average scores over a set of semantically similar words. This semantic information is found using an automatically constructed (unsupervised) thesaurus. This approach outperforms another unsupervised approach using the more principled Maximum Entropy approach (see below), and we believe that the use of maxent with very large, noisy datasets is a promising avenue for future research.

2.5.2 Loglinear and Maximum Entropy classifiers

Loglinear or exponential models, of which maxent is a special case, offer a completely different way of modelling the conditional distribution $p(c|x)$. Chapter 3 covers maxent models in more detail, but for convenience here is a short summary. The general form is a product of exponential factors (hence the name):

$$p(x) = \pi \prod_i \alpha_i^{f_i} \tag{2.15}$$

where π is a normalisation constant, each f_i is a feature and α_i is an associated weight. Features are simply questions about the data, for example:

$$f_i = \begin{cases} 1 & \text{if attachment = 1, preposition = } \textit{of} \\ 0 & \text{otherwise} \end{cases}$$

Loglinear models offer a general way to combine diverse information sources into a single model. The weights are set to maximise both the likelihood of the data (as with maximum likelihood estimation) and also the entropy of the conditional distribution.

2.5.2.1 Supervised Maxent: Ratnaparkhi, Reynar and Roukos 1994

Ratnaparkhi et. al. were the first to apply maxent to the PP disambiguation problem, and the supervised dataset created for this project has subsequently become the de-facto standard for this task. This approach uses two pools of features: “word” features querying the presence of any n-tuple of headwords ($n <= 4$); and “class” features — all words in the dataset were assigned to classes using the mutual information clustering algorithm [Brown et al., 1992].

The set of features used in the model is selected greedily using an incremental selection algorithm. This algorithm starts with a small pool of single-word features, and incrementally adds selected features to the model and more complex features to the pool. The best models achieved a final accuracy of around 81.6%. Crucially, these models still only have around 200 features, a limit perhaps imposed by the computer technology of the day. In Chapter 4 we find that a maxent model with a much larger feature set, using as many as 172,222 features, achieves an accuracy of 84.4%. Although features can potentially be quite complex it appears that many are composed of just one or two words.

2.5.2.2 Loglinear Model: Franz 1996

Franz uses a loglinear classifier with three types of features:

- *Preposition* features query the presence of a particular preposition.
- *Noun-level* features measure the lexical association strength between a particular subject noun (N_1) and a preposition.
- *Verb-level* features do the same, but for verbs and prepositions.

The lexical association strength is a measure of how frequently two words occur together. This has echoes of Hindle & Rooth’s lexical preferences, although Franz uses a mutual information score rather than word frequencies. Other features were considered but not included; these include the POS tags for the preceding noun and the prepositional noun, and whether the preceding noun was preceded by a definite article.

The final classifier achieved an accuracy of 82%, but the dataset is unique to this project so only rough comparisons can be made to other approaches. As with Hindle & Rooth's approach, this approach does not consider the prepositional noun (N_2). The loglinear framework used in this approach also does not necessarily consider interactions between all features, as the maxent framework used in this dissertation does.

2.5.2.3 Unsupervised/Supervised Maxent: Ratnaparkhi 1998

There are two maxent approaches presented in this thesis. The first is unsupervised insofar as there is no manual labour or full parser involved in the extraction of the training data. Instead, the raw text is processed using a tagger and a simple chunker[1] to determine the headwords of noun and verb phrases. A heuristic is used to extract unambiguous prepositional phrases and these are used to train the model.

The classifier uses a model of the joint distribution $P(A,V,N_1,P,N_2)$ which is factored as:

$$p(A,V,N_1,P,N_2) = p(V)P(N)p(A|V,N_1)p(P|A,V,N_1)p(N_2|P,A,V,N_1) \qquad (2.16)$$

The pivotal term is $P(P|A,V,N_1)$, the probability of a particular preposition given a particular attachment position, a verb and a noun. The author presents three different ways to model this distribution, two of which use maximum likelihood-based estimation and the third using maxent. For the maxent model, the feature space consists of all (V,P) and (N_1,P) tuples, all P's, $f_N(p)$ and $f_V(p)$: the number of times that each preposition occurred with a noun or a verb. Surprisingly, the best approach uses just raw counts and the overall classifier scores 81.91% accuracy. When the maxent model

[1] A chunker is a simpler kind of parser that attempts to group words into phrases but does not find associations between phrases

is included, the classifier scores 81.82% accuracy. Ratnaparkhi suggests that this could be improved with a more accurate extraction process.

The second supervised approach uses the benchmark dataset and compares different feature selection algorithms. The feature set consists of the following word combinations:

1. 1-Tuples: (V), (N_1), (P)
2. 2-Tuples: (V,P), (N_1,P), (P,N_2)
3. 3-Tuples: (V,P,N_2), (N,P,N_2), (V,N_1,P)
4. 4-Tuples: (V,N_1,P,N_2)

All preposition-less tuples except for (N_1) and (V) are omitted. The three feature selection algorithms, along with the resulting accuracy figures, are:

1. Global frequency cutoff: only those features occurring more than c times are retained. In other words, if $c = 1$ then any features occurring once are discarded, and if $c = 0$ then all features are kept[2]. Using $c = 1$ this model has 4,028 features and achieves an accuracy of 82.0%.
2. Tuned frequency cutoff: as above, but c is parameterised by tuple size, so tuples of different sizes are allowed to have different cut-offs. Here $c_1 = 9$, $c_2 = 1$, $c_3 = 0$ and $c_4 = 0$. This model has 8,875 features and an accuracy of 83.7%.
3. Incremental feature selection: The maxent framework itself is used to incrementally select features that lead to the largest expected increase in the likelihood of the training data (see the following chapter). This model selects a mere 387 features but the accuracy is only 80.5%.

We find in this dissertation that preposition-less tuples eliminated *a priori* here do have a contribution to make, and our maxent model using the same global cut-off but with all possible tuples achieves 82.6% accuracy. We also find that maxent models in this

[2]This numbering scheme is different to the one used in the paper, where any features occurring fewer than c times were discard. To convert from that system to this one, simply subtract one.

domain are very sensitive to the value of c and the values used by Ratnaparkhi are generally too high. Using the same feature set as this approach but with a global cut-off rate $c = 2$, we obtain a model with twice the number of features (8,611) but with 83.5% accuracy. Finally, by applying some morphological processing to the dataset, using a model with all possible tuples and reducing the cut-off rate to $c = 1$ we produce a classifier with a significantly higher 84.4% accuracy rate[3].

2.5.3 Other approaches

The following sections list a variety of other approaches in chronological order.

2.5.3.1 Transformation-Based Learning: Brill and Resnik 1994

Transformation-based error-driven learning is a supervised method that starts with some heuristic rule for assigning classes to examples. We expect that this initial classification rule will produce a number of errors, and to fix these errors we supply the algorithm with a set of new rules for changing the classifications. In the case of PP classification, a transformation rule might be:

```
If V is ``have'' and P is ``in'' then change attachment location to
``V''
```

The algorithm chooses the rule that leads to the largest net reduction in errors. This step is repeated, with the algorithm selecting new rules that fix the largest number of remaining errors, until some stopping criterion is reached. The model classifies a new dataset by first applying the initial rule to all items, then reclassifying these predictions by applying each rule in the chosen order.

The set of allowable rules from is defined using rule templates. For example, the template for the above rule would be:

```
Change attachment to A if V is W1 and P is W2
```

[3]All statistical tests in this dissertation are done using McNemar's test [Dietterich, 1998] at a 5% confidence level

This approach considers templates using questions about any combination of three or fewer headwords. Subsequent experiments used WordNet [Miller, 1990], allowing questions about which semantic class a noun belonged to.

The final performance of this system is 81.8% when semantic classes are used, on a supervised dataset drawn from the same corpus as the Ratnaparkhi dataset but with only half the number of examples. Transformation-based learning does produce models with very few parameters: the best model used around 266 rules, and the authors argue that the final model is also more transparent than statistical models. Nevertheless, this accuracy is significantly lower than many other approaches using lexical and semantic information.

2.5.3.2 Decision Trees and WordNet: Stetina and Nagao 1997

This approach combines a word-sense disambiguation algorithm with decision trees to achieve 88.1% accuracy on this task, significantly higher than all other approaches and on a par with human-level accuracy. Decision trees have previously been used by [Ratnaparkhi et al., 1994] on this task with quite poor results. The key innovation of this paper is to assign semantic senses to each word using a word-sense disambiguation algorithm based on WordNet. This resolves the spares data problem by grouping together words with similar attachment preferences. For example, consider the following two examples taken from the paper:

(2.17) Buy books for children

(2.18) Buy magazines for children

If *books* occurred in the training data but *magazines* did not, then we can guess the attachment site for the second sentence using our estimate for the first. The word-sense disambiguation algorithm uses a measure called *semantic similarity*. The sense assigned to each word depends on how frequently that sense occurs in similar contexts in the dataset. For example, given the following two sentences (also taken from this paper):

(2.19) The investor bought the company for 5 million dollars

(2.20) The investor purchased the company for 5 million dollars

then this algorithm will assign the sense *purchase* to the verb buy (other possible senses include *bribe* and *believe*) since this reading is consistent with other examples in the dataset. The semantic distance measure used to define similarity is based on the distance between the two senses in the WordNet hierarchy. The results achieved by this approach are surprisingly high given the previously poor performance obtained using decision trees; also, no other approach has achieved such significant gains by adding semantic information using WordNet. This approach is not directly comparable to our own best results, since we aim to limit the amount of supervision in our classifier to the training set alone.

2.5.3.3 Memory-Based Learning: Zavrel, Daelemans and Veenstra 1997

Most of the learning algorithms described thus far induce a classifier given some training data but then throw away the data. Memory-based learning (MBL) essentially does the opposite: it keeps all the (supervised) training data, and produces classifiers on demand. Each training item is seen as a point in the space of all possible feature values (in this case, the possible headwords). After storing all training items along with their class labels, each new unseen item is compared to its nearest neighbours in feature space and assigned the most likely class label. This approach is potentially useful if the data has many exceptional cases or irregularities that are difficult for an induced classifier to model.

MBL is an explicitly supervised algorithm and this paper uses the standard Ratnaparkhi dataset. The final model using purely lexical features achieves 84.1% accuracy. The authors also make a stab at the unknown word problem and generalisation by using the Lexical Space clustering algorithm. Here each word is replaced with a vector “fingerprint” based on its co-occurrence statistics gleaned from a large corpus. Incorporating these features yielded a slightly improved accuracy of 84.4%. The training corpus for the Lexical Space algorithm was in fact the entire WSJ, ensuring that unknown words would not be a problem for these features.

2.5.3.4 AdaBoost: Abney, Schapire and Singer 1999

AdaBoost, and boosting algorithms in general, produce collections of classifiers that together vote on the correct prediction [Schapire, 1999]. AdaBoost takes a set of supervised training examples and assigns a weight to each. These weights can be thought of as artificially adjusting the frequency of each example. In each iteration of training, a single classifier is selected and tested on the training set. The weights are then updated so correctly classified examples are weighted more and correctly classified examples are weighted less. The final prediction is the weighted vote of each selected classifier, where the weight is derived from the accuracy of the classifier on the training set. AdaBoost can be also be seen as combining classifiers to minimise the cost of misclassified examples [Mason et al., 2000]. This is in contrast to maximum likelihood methods that estimate the distribution $p(A|X)$ and then use it to classify new examples.

For this task, each classifier is a simple predicate of the form "attachment = A if context = X" where the context can be any subset of the PP 4-tuple. Using the Ratnaparkhi dataset with some morphological processing there are 166,683 distinct combinations of four or fewer headwords; the final model used 9,677 of these after 20,000 training iterations. The final accuracy is 84.4%, equalling the best models using no additional supervision (Ratnaparkhi et. al., Brill & Resnik, Zavrel et. al).

2.5.3.5 Neural Networks: Alegre, Sopena and Lloberas 1999

Artificial neural networks are a general class of learning algorithms inspired by the structure of biological brains. The standard supervised feed-forward neural network as used in this approach learns a non-linear function defined by a set of inputs and outputs. Categorical attributes such as words are usually encoded using a local, 1-of-*n* system required a separate input for each possible category. Unfortunately neural networks do not scale well to very large input spaces, and are therefore not commonly seen in natural language applications. Alegre et. al. get around this problem by using

WordNet semantic classes and other generalisations in place of actual words. There are eight different feature categories in this approach:

- All applicable semantic classes for N_1, N_2 and V (see below).
- The 36 most common prepositions.
- The set of possible prepositions a verb can subcategorise. For example, *accuse* usually subcategorises *of*.
- The set of possible prepositions that can occur with a particular prepositional noun phrase, as determined by the training set.
- The root nodes of the 15 different verb hierarchies defined by WordNet.
- Two binary features returning true if N_1 or N_2 respectively are proper nouns

WordNet has a large number of possible noun classes, but in this approach only those classes required for the training and test sets were coded for. It is not clear whether this approach scales up to the more general case. Using these features, even a linear mapping from inputs to outputs (a perceptron) has a published accuracy rate of 83.08% accuracy, a surprisingly high result. The usual Ratnaparkhi dataset is used for training and testing, although an additional 12,029 examples were used for validation and for selecting the best neural net. An ensemble of neural nets (see Chapter 6 for more information about ensemble methods) achieved 86.08%. This approach achieves slightly higher results than our maxent classifier, but at the cost of additional supervision in the form of WordNet.

2.6 Chapter summary

Two recurring themes in this review have been the decision about which classifier to use and which features and tuples to include in that classifier. Chapter 5 considers several new feature types, and Chapter 6 compares three general methods for selecting features from a large pool of candidates. Chapter 4 is essentially about the choice of classifier: [Collins and Brooks, 1995] achieve very good results using a back-off algorithm, but the underlying model – maximum likelihood – has several shortcomings. Maximum entropy models have several attractive properties, and it seems quite possible to improve on the [Ratnaparkhi et al., 1994] and [Ratnaparkhi, 1998] maxent models. We consider both vanilla and backed-off maxent models. Ensemble methods, where a number of classifiers vote on the final prediction, have essentially been untouched in this field except for the Alegre (1999) approach. Chapter 8 considers some methods for creating ensembles of maxent classifiers.

Chapter 3

Maximum Entropy

3.1 Introduction

As described in Section 2.5.1, we can create a PP classifier by modelling the conditional probability distribution $p(A|X)$. Section 2.5.1 introduced maximum likelihood estimation:

$$\tilde{p}(a|x) = \frac{f(a,x)}{f(x)} \tag{3.1}$$

This maximises the likelihood of the training data (hence the name) but in some cases this is not really what we want: we have the problems of sparse data and noisy estimates. Smoothing schemes can resolve these problems to some extent, but these schemes are often ad-hoc and heuristic in nature. Ideally we would like an estimation procedure that only fits those parts of the dataset we believe to be true and also leaves some probability for unseen events. Both of these desiderata are fulfilled by the maximum entropy estimation procedure presented in this chapter.

Maximum entropy has an excellent pedigree in natural language processing: in addition to the PP attachment approaches considered in the previous chapter, Berger (1996) has applied it to machine translation, Ratnaparkhi (1996) used it for tagging and sentence boundary identification, Charniak (1999) incorporated maxent models in a parsing system and several authors have used maxent for language modelling (eg [Lau, 1994, Rosenfeld, 1994]).

3.2 The maximum entropy principle

Entropy has many interpretations: it is often used as a measure of randomness and disorder, where high entropy indicates extreme randomness. For our purposes it can be thought of as a measure of uniformity. In the absence of any other information, this maximum entropy distribution seems an intuitively right one to use — imagine predicting the outcome of a coin toss without knowing about the coin or guessing about prepositional phrase attachment in ancient Sumerian. The conditional entropy $H(A|X)$ of the distribution $p(A|X)$ is defined to be

$$H(A|X) = -\sum_{c,x} p(a,x) \log p(a|x) \tag{3.2}$$

This is maximised when the distribution is uniform over all classes, meaning that all classes are equally likely given any example. The principle of maximum entropy [Jaynes, 1957] states that:

> ... in making inferences on the basis of partial information we must use the 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.

3.3 Using features to encode our partial knowledge

Knowledge is encoded in the form of *features* and *contextual predicates*[1]. A contextual predicate is a boolean function that returns true if the current example contains

[1] This terminology is borrowed from [Ratnaparkhi, 1998]

a context x. For example, we might want to encode the attachment preference of the quadruple (eat,pizza,with,anchovies). The contextual predicate we use is:

$$cp(x) = \begin{cases} True & \text{if context = “eat pizza with anchovies”} \\ False & \text{otherwise} \end{cases}$$

These predicates are the basic building blocks of a maxent model. A feature itself is a non-negative numeric observation about a contextual predicate x and an attachment a. In this dissertation, we limit ourselves to binary functions that return 1 (true) if context x belongs to the attachment class a and 0 (false) if not:

$$f(a,x) = \begin{cases} 1 & \text{if A = VA, x = “eat pizza with anchovies”} \\ 0 & \text{otherwise} \end{cases}$$

Each feature $f(a,x)$ has an expected value $E[f]$ over the training set:

$$E[f] = \sum_x q(a,x) f(a,x) \tag{3.3}$$

where $q(a,x)$ is the observed probability of an (a,x) tuple in the training set. Because we are using binary features, $E[f]$ is equivalent to the probability of (a,x) occurring in the training set. We can also calculate the expected value of a feature in our maxent model:

$$E[f] = \sum_{a,x} q(x)\tilde{p}(a|x) f(a,x) \tag{3.4}$$

where $\tilde{p}(a|x)$ is the conditional probability of a in our model. The maxent model is constrained so the expected value of each feature in model is the same as its expected value in the training set:

$$\sum_{a,x} q(a,x) f(a,x) = \sum_{a,x} q(x)\tilde{p}(a|x) f(a,x) \tag{3.5}$$

For example, consider our well-worn pizza examples. Given a feature:

$$f(a,x) = \begin{cases} 1 & \text{if A = VA, x = “eat pizza with”} \\ 0 & \text{otherwise} \end{cases}$$

we may know that $E[f]$ = 0.5. There are two equally probable matching examples x_1, x_2 in the training data, so there are (infinitely) many ways to match the constraints in (3.5):

- $p(VA|x_1) = 0.2$, $p(VA|x_2) = 0.3$
- $p(VA|X_1) = 0.1$, $p(VA|x_2) = 0.4$
- $p(VA|X_1) = 0.25$, $p(VA|x_2) = 0.25$
- ...

However, only the last model also satisfies our maximum entropy requirement by being as uniform as possible.

3.4 Log-linear models

It can be shown ([Ratnaparkhi, 1997], Theorems 1 and 2) that a model satisfying the constraints (3.5) and also maximising entropy must use have the following log-linear (exponential) form:

$$\tilde{p}(a|x) = \frac{1}{Z(x)} \prod_{i=1}^{n} \alpha_i^{f_i(a,x)} \tag{3.6}$$

where Z is a normalising factor ensuring the the distribution sums to 1:

$$Z(x) = \sum_{a} \prod_{i=1}^{n} \alpha_i^{f_i(a,x)} \tag{3.7}$$

The particular form of model that we use here is:

$$p(a|x) = \frac{1}{Z(x)} \exp\,(\sum_{i=1}^{n} \lambda_i f_i(a,x)) \quad (3.8)$$

$$Z(x) = \sum_{a} \sum_{i=1}^{n} \exp\,(\lambda_i f_i(a,x)) \quad (3.9)$$

Each λ_i is a free parameter for a single feature f_i chosen to ensure that the constraints hold. Technically the parameters are Lagrange multipliers used to solve a constrained optimisation problem, but they have a more intuitive interpretation as *importance weights* for each feature. If the feature $f_i(a,x)$ is highly informative then the associated weight λ_i will be high.

3.5 Improved Iterative Scaling

Finding suitable weights analytically is impossible except for very small models, so instead an iterative procedure is used. Two of the most popular are the Generalised Iterative Scaling (GIS) [Darroch and Ratcliff, 1972] and Improved Iterative Scaling (IIS) [DellaPietra et al., 1997] algorithm. The IIS algorithm, used in this dissertation, is tuned for maximum entropy models and allows for positive-valued features, whereas GIS requires features to have binary values.

First some preliminaries[2]. As described above, $q(a,x)$ is the observed probability of a set of (a,x) pairs in the dataset. The likelihood of this same data being produced by some model $\tilde{p}(a,x)$:

$$L_{\tilde{p}} = \prod_{a,x} \tilde{p}(a|x)^{q(a,x)} \quad (3.10)$$

The model that maximises this is as close to the empirical distribution as possible given the model framework. Of course, as described above, this is not necessarily what we want. The key advantage of using maximum entropy models is that we get to specify

[2]This treatment is adapted from the excellent introductory paper [Berger, 1997]. An equivalent view using Kullback-Liebler divergence is discussed in [Ratnaparkhi, 1997]

exactly which parts of the empirical distribution that we want to match — namely, those parts that we believe to be true or have reliable data for. The product in (3.10) is awkward to deal with, so a more common measure is *log-likelihood* with a more manageable sum:

$$L_{\tilde{p}} = \sum_{a,x} q(a,x) \log \tilde{p}(a|x) \tag{3.11}$$

Substituting in our conditional model using (3.9) gives us:

$$L_{\tilde{p}} = \sum_{a,x} q(a,x) \sum_{i} \lambda_i f_i(a,x) - \sum_{x} q(x) \log \sum_{y} \exp(\sum_{i} \alpha_i f_i(a,x)) \tag{3.12}$$

Differentiating this with respect to a particular parameter α_i gives us:

$$\frac{\partial L_{\tilde{p}}}{\partial \alpha_i} = \sum_{a,x} q(a,x) f_i(a,x) - \sum_{a,x} q(x) \tilde{p}(a|x) f_i(a,x) \tag{3.13}$$

The right and left-hand sides of this last equation are the same constraints on features expressed in (3.5), that is, that the expected value of f_i in the empirical distribution should match the expected value of f_i in the conditional model. But using the maximum likelihood approach we now have the means to find a solution.

Finding the model that satisfies (3.12) is a constrained optimisation problem with n constraints and n parameters. An attractive property of log-linear models is that there is only one possible solution that satisfies the constraints, and this solution will also be the one that maximises entropy. Happily, it also turns out that there is only one such solution and it lies at the extremum of a convex function. This means that a hill-climbing strategy (such as IIS) is guaranteed to eventually converge on the desired answer.

The key to the IIS algorithm is a growth function that calculates a change for each feature weight that will increase the log-likelihood of the data, unless the model has converged, in which case there will be no change. If δ_i is the change to the weight α_i, then the IIS growth function is:

$$\sum_{a,x} q(a,x) f_i(a,x) - \sum_x q(x) \sum_y \tilde{p}(a|x) f_i(a,x) e^{\delta_i f^{\#}(a,x)} \tag{3.14}$$

where $f^{\#}(a,x) = \sum_i f_i(a,x)$. For binary functions of the kind used here, $f^{\#}$ is simply the number of "active" features for a given example. If $\tilde{p}(\lambda)$ is the original model and Δ is the vector of weight changes, then:

$$L_{p(\tilde{\lambda+\Delta})} \geq L_{p(\tilde{\lambda})} \tag{3.15}$$

In other words, the log-likelihood of the new model is greater than or equal to the log-likelihood of the original model, and if the two are equal then the model has converged. The IIS algorithm therefore proceeds as follows:

- Set the initial weights $\lambda_i = 0$
- For each weight:
  - Find the unique solution to (3.14).
  - Set each weight $\alpha_i \leftarrow \lambda_i + \lambda_i$
- Stop if $\tilde{p}(a|x)$ has converged or some other stopping criterion has been reached

There is usually no need to run IIS until convergence, in fact this may be a bad idea since some weights can grow without bound (see below). Common strategies are to iterate until there is no significant increase in accuracy on a held-out set, or simply until some fixed number of iterations has been performed. In this domain we find that useful models are found very quickly: even the largest models with 200,000+ parameters only need 15-20 training iterations.

3.6 Maxent and overfitting

Maxent models have an inherent preference for smoother models, but are susceptible to overfitting particularly with low-frequency features. For example, consider a feature that only considers the noun and a particular noun only occurs once in the dataset – *charity* perhaps, given the source of the Ratnaparkhi dataset. If this sole occurrence is a case of verb attachment, then the constraint placed on the model is:

$$\sum_a q(a, charity) = q(charity) \sum_a \tilde{p}(a|charity) \tag{3.16}$$

Since $q(A = VA, charity) = 1$, the scaling algorithm will push $\tilde{p}(1|charity)$ towards one by increasing the corresponding feature weight. But in an exponential model the probability $\tilde{p}(0|charity)$ can never go to zero so the weight for its counterpart feature grows without bound.

3.7 Choosing features

Nothing has been said so far about which features to include in a model. There are usually three stages:

1. Choose feature *types* or *templates*. This decision must be made *a priori* by the model designer. For example, if we decide that prepositions are important, then we add the feature template:

$$f_i(a,x) = \begin{cases} 1 & \text{if A = VA, Preposition = P} \\ 0 & \text{otherwise} \end{cases}$$

2. Identify the set of candidate features. This is done by creating all possible instantiations of the feature templates in a dataset.

3. Feature selection. The final stage is deciding which actual features to include the final model. Not all features may be included in the final model because of the overfitting problem and simply a preference for smaller models.

Features are allowed to interact, meaning that the value of one feature need not be independent of other features.

3.8 Applying maxent to PP attachment

Following in the footsteps of [Ratnaparkhi, 1998], we can construct a maxent model to resolve PP ambiguity. As a sanity check, we first construct a model using exactly the same features and frequency cut-off points as the supervised maxent model described in that paper. The features are implemented as described above; for example the tuple (v,n_1) for the pizza sentence gives the feature:

$$f(a,x) = \begin{cases} 1 & \text{if A = VA, V = "eat", N}_1\text{ = "pizza"} \\ 0 & \text{otherwise} \end{cases}$$

Our model[3] achieves an accuracy of 82.22%, a stone throw from the 82.0% result published in that paper.

However, as mentioned in the previous chapter, the frequency cut-off is a sensitive parameter in this domain and those used in the Ratnaparkhi model is too harsh. Using all possible feature types, including those without a preposition, and a global cut-off $c = 1$ raises accuracy to 84.17%. Using the same model but applying some morphological processing to the dataset (see Section 4.3.1) produces a classifier with 84.44% accuracy and 25,777 features. Tuning the cut-off parameters further raises accuracy to 84.79% (see Chapter 5). This means maxent now compares favourably with Memory-Based Learning, Boosting and back-off models for this task. Despite the large number of parameters, the mode is still fast to train and test, with the complete cycle taking around two minutes on a Sun Ultra5 360 Mhz workstation.

[3]The maxent software used in this dissertation was written by Rob Malouf of the Rijksuniversiteit Groningen and can be downloaded from http://www.let.rug.nl/˜malouf/software/iis.tar.gz

3.9 Chapter summary

To summarise, maximum entropy models have the following advantages:

- We can include any kind of knowledge that can be expressed as a positive numeric feature.
- The model makes no additional assumptions beyond what we specify in our features.
- Maxent models interaction and dependence between features.
- If the constraints are consistent then there exists a unique solution which can be found iteratively using the IIS algorithm.

This dissertation also shows that maxent models also achieve state-of-the-art accuracy on this domain compared to other approaches using purely lexical features.

A disadvantage of maxent models is their potential to overfit the data, so we need some way of identifying only those features that will be useful and we reliable estimates for. Chapter 5 considers compares frequency cut-offs with two other feature selection methods: information gain and weight variance.

Chapter 4

Maxent and Linear Interpolation

4.1 Introduction

This chapter presents our main results using standard lexical features with a maxent model. This classifier uses simple combinations of the standard benchmark PP 4-tuple, but it achieves 84.71% accuracy — on a par with the Collins & Brooks (CB) backing-off model, Memory-Based Learning and AdaBoost. Following in the footsteps of the CB model, we also present a backed-off maxent model that achieves 84.63% accuracy. Backing-off is a specific case of linear interpolation, where the final prediction is the weighted combination of a number of models. The weights can be set with respect to a held-out dataset, allowing a linear interpolation model to be smoothed and moved away from the maximum likelihood estimate. However, a linear interpolation of classifiers may make suboptimal use of the individual classifiers. These results show that a single maxent model has equivalent generalisation power of a linear interpolation model, despite only having access to the training data.

4.2 Linear Interpolation

Linear interpolation is a general method for combining models where the output is a weighted sum of the component models. It is often used as a means of overcoming the sparse data problem endemic in natural language applications. A model of the distribution $p(A|V,N_1,P,N_2)$ should be accurate since it uses a large context but it will have no estimates for many (V, N_1, P, N_2) tuples. Using a model with fewer context words such as $p(A|V,N_1,P)$ partly solves the sparse data problem but this model is unlikely to be as accurate. Linear interpolation offers a way around this dilemma by combining both estimates into a single prediction:

$$\tilde{p}(A|V,N_1,P,N_2) = \lambda_1 p(A|V,N_1,P) + \lambda_2 p(A|V,N_1,P,N_2) \tag{4.1}$$

where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$, ensuring that the result is a proper probability distribution. Of course, the choice of the two models in (4.1) is purely arbitrary; we could have used any models with any selection of context words.

In (4.1) the weights are set globally. A generalisation of this scheme is to parameterise the weights by context. This allows us, for example, to place more weight on the 4-tuple model if we have seen a particular 4-tuple many times. Finding appropriate values for all of these new parameters does place additional strain on our already sparse data, so weights are usually grouped together in some way. For example, the CB model is a special case of linear interpolation where the weight of a particular model $\lambda_i(x) = 1$ only if that model would have been chosen by backing-off, otherwise $\lambda_i(x) = 0$.

4.3 The baseline model: using single-word features

We begin with a baseline model composed of single-word features. A single-word feature queries a specific word in a specific position, for example:

Iterations	Mean $f(x) = 1$	SD	Mean $f(x) > 1$	SD
15	0.224	0.464	0.768	0.290
100	1.535	1.943	0.485	2.252
500	2.323	5.707	0.767	6.281
1000	2.691	8.320	0.900	9.096

Table 4.1: Feature weights for low and high frequency features

$$f_i = \begin{cases} 1 & \text{if Attachment = 1, Verb = ``buy''} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

There are 17,932 unique single-word features given the training data. Some features only occur once, for example:

$$f_i = \begin{cases} 1 & \text{if Attachment = 1, Verb = ``gauges''} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

but many occur several times, like (4.2) above. The baseline model uses all possible features, although the following chapter shows that being more selective produces a more accurate model. The accuracy of this full model is 81.50%, on a par with the [Ratnaparkhi et al., 1994] maxent model, despite only considering very small contexts.

As described previously in Section 3.6, maxent models are susceptible to overfitting. According to Mitchell (1997), a model overfits if a smaller model has a higher accuracy rate on the set of all examples but the larger model has a higher accuracy on the training set. A textbook case of overfitting, then, can be seen in Figure 4.1. This graph plots the accuracy of a full model and a cut-off model ($c = 2$) on a validation set (used as an approximation of the entire set of examples). The accuracy of the larger model on the training set improves monotonically, but the accuracy on the validation set drops almost immediately. Table 4.1 shows mean feature weights for high and low frequency words. As described in Section 3.6, we have reason to believe that very high feature weights are due to overfitting rather than highly informative features.

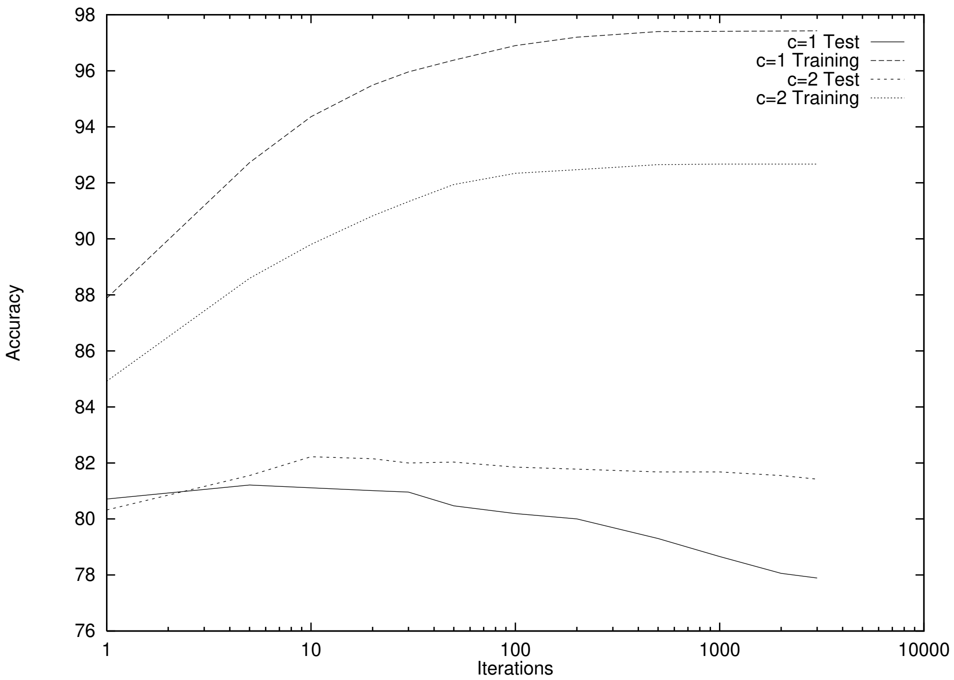


Figure 4.1: Accuracy results for the baseline model as a function of training iterations. Note the log scale along the x axis.

Processing	Features	ValSet	XVal (SD)
None	17932	81.11	81.64 (0.84)
Years	17866	81.28	81.67 (0.85)
+ Case	17754	81.26	81.68 (0.88)
+ Numbers	16897	81.38	81.80 (0.92)
+ Names	15240	81.23	81.73 (0.97)
+ Stemming	12972	81.41	81.99 (1.07)

Table 4.2: A step-by-step analysis of the morphological filters used in this dissertation

4.3.1 Morphological processing to reduce the sparse data problem

A partial solution to the sparse data problem is to reduce the number of possible contexts by grouping similar words together. For example, we might believe that the morphological stem of a word (eg *eat*) will behave in the same way as its derived forms (*eating*, *ate*). That is, we believe that for all words w, $p(A|stem(w)) \approx p(A|w)$. Not only do we effectively increase the number of known words, but we improve our estimates of those words. We use the following steps, based on those used in [Collins and Brooks, 1995], to make the data a bit less sparse:

- All 4-digit numbers are replaced with the string "YEAR"
- All digits are replaced with the string "NUM".
- All verbs and prepositions are converted to lower case.
- All n_1 and n_2 words starting with an upper case letter followed by one or more lower case letters are replaced by the token "NAME".
- All nouns are stemmed using the PCKIMMO morphological analyser [1], a variant of that used in [Collins and Brooks, 1995].

The results of progressively applying these processing steps to the single-word model are shown in Table 4.2. The steps are ordered according to how many words are affected by each step. Two accuracy figures are given, the first for the validation set (ValSet) and the second for ten-fold cross-validation over the entire dataset (XVal).

These results are hardly a resounding vote in favour of morphological processing. Applying the “Names” step here looks particularly dubious. The baseline model model trained without using the “Names” step achieves 81.45% accuracy on the validation set. Applying this morphological processing usually doesn’t harm the model greatly, and using all of these steps also allows us to compare our model directly with the CB model, the subject of the next section. The final accuracy of a single-word model with full morphological processing on the test set is 82.08%, an improvement of 0.58% over the baseline model.

4.4 Using all features

We now consider the more general case where a feature can use any non-empty subset of the PP 4-tuple. There are 15 distinct subsets, but the features can be divided into four groups: 4-tuple features pose questions about all four words, 3-tuple features ask about a subset of three words and so on. Note that we consider all tuples, even those lacking a preposition. An example of a 3-tuple feature could be:

$$f_i = \begin{cases} 1 & \text{if Attachment = VA, Verb = “eat”, Noun}_1 \text{ = “pizza”, Preposition = “with”} \\ 0 & \text{otherwise} \end{cases}$$

Given the usual training set, there are 197,448 possible features before morphological processing and 172,222 after. Most of these are either 2-tuple or 3-tuple features (41% and 40% respectively). The question explored in this section is how these features can be combined. The first strategy is linear interpolation, based on the CB model. The second strategy includes all possible features in a single model.

[1]This is freely available from http://www.sil.org/pc-kimmo/pc-kimmo.html

4.4.1 In a linear interpolation model

This section presents classifiers that combine multiple models using linear interpolation. As described in the introduction, this definition includes backing-off models such as the CB algorithm. The maxent version of this algorithm has four progressively simpler models $p_{4..1}$, where the model p_n only uses n-tuple features. This back-off algorithm mirrors that of Collins & Brooks:

1. if $f(4-tuple) > c_4$ then $p(1|v,n_1,p,n_2) = \tilde{p}_4(1|v,n_1,p,n_2)$
2. else if $f(3-tuples) > c_3$ then $p(1|v,n_1,p,n_2) = \tilde{p}_3(1|v,n_1,p,n_2)$
3. else if $f(2-tuples) > c_2$ then $p(1|v,n_1,p,n_2) = \tilde{p}_2(1|v,n_1,p,n_2)$
4. else if $f(1-tuples) > c_1$ then $p(1|v,n_1,p,n_2) = \tilde{p}_1(1|v,n_1,p,n_2)$
5. else attachment = 1 (the default attachment, this is never used)

where c_i determines the threshold at which the model p_i is called into action and $f(n$-tuple$)$ is a shorthand for the summed frequency of all combinations of size n. In contrast to the CB model, we consider all possible word combinations, so

$$
\begin{aligned}
f(1-tuples) &= f(v)+f(n_1)+f(p)+f(n_2) \\
f(2-tuples) &= f(v,n_1)+f(v,p)+f(v,n_2)+f(n_1,n_2) \\
\ldots & \quad \ldots
\end{aligned}
$$

The preposition-less tuples were excluded from the CB model since they carried much less predictive power and harmed the accuracy of the model. But the maxent scaling algorithm weights the less informative features appropriately, and we find that excluding the preposition-less tuples harms the accuracy of the model. The optimal setting is $c_i = 0$ for all i, but as in the original paper, stages 1 and 2 are not used if $\tilde{p}(1|v,n_1,p,n_2) = 0.5$.

Table 4.3 summarises the performance of this model and compares it to the CB model. A cutoff of $c = 2$ is applied for 1- and 2-tuples and $c = 1$ for 3- and 4-tuples. The

	Maxent		Back-Off	
Model	Selected	Accuracy	Selected	Accuracy
4-Tuples	160	96.88	243	93.0
3-Tuples	686	89.65	982	87.6
2-Tuples	1808	84.02	1735	82.5
1-Tuples	443	73.14	134	73.1
Default	0		3	100.0
Total	3097	83.40	3097	84.53

Table 4.3: This table compares a backed-off maxent model with frequency cut-offs with the CB model

overall accuracy is comparable with its rival, showing that for this domain, the maxent smoothing scheme compares favourably to the simpler smoothing scheme used in the CB model.

We can generalise this algorithm by allowing nonzero weights for any applicable models at each backing-off stage. For example, if $f(4-tuples) = 0$ but $f(3-tuples) > 0$ then the final prediction is a weighted sum of the four possible models:

$$P(1|v,n_1,p,n_2) = \lambda_3 \tilde{p}_3(1|3-tuples) + \lambda_2 \tilde{p}_2(1|2-tuples) + \lambda_1 \tilde{p}_1(1|1-tuples) \tag{4.4}$$

If all but the most specific λ_i is set to zero, then we again have the back-off algorithm described above. The weights were set semi-automatically using the validation set, and Table 4.4 shows the results. The interpolated model achieves an accuracy of 83.53%, a slight improvement over a straight back-off model.

This kind of interpolation can also be used with the Collins & Brooks model, but the original weight setting is optimal. Interestingly, applying a cut-off to the CB model also improves accuracy slightly. setting $c = 2$ for 3-tuples, 2-tuples and 1-tuples and $c = 1$ (no cut-off) for 4-tuples creates a model with 84.63% accuracy.

Model	λ_4	λ_3	λ_2	λ_1	Accuracy
$f(4-tuple) > 0$	0.95	0.05	0.0	0.0	96.88
$f(3-tuples) > 0$		0.7	0.2	0.1	89.80
$f(2-tuples) > 0$			0.8	0.2	84.40
$f(1-tuples) > 0$				1.00	73.14
Total					84.63

Table 4.4: The results of a more general interpolated maxent model

4.4.2 In a single model

The simpler alternative to interpolated models is including all possible features in a single model. This means that the single model has access to more information, but interpolated models have the advantage of being tuned against a validation set. Despite this potential handicap, we find that single models show slight (although not significant) improvements over an interpolated model and are also simpler to train and use.

As with previous work using maxent models, we can impose frequency cut-offs globally or locally. The optimal global cut-off is $c = 1$, that is, any feature only occurring once is discarded. This model achieves 84.44% accuracy. A tuned model achieves slightly better accuracy rates. The optimal cut-off rates are found to be:

- 4-Tuples: $c = 0$
- 3-Tuples: $c = 1$
- 2-Tuples: $c = 1$
- 1-Tuples: $c = 1$

Table 4.5 shows a stage-wise analysis of both models. In this case, a stage is “selected” for a given model if the most complex available feature belonged to that stage.

	No Cut-off		Global		Local	
Model	Selected	Accuracy	Selected	Accuracy	Selected	Accuracy
4-Tuples	243	93.00	160	96.88	243	93.00
3-Tuples	1003	88.43	686	90.38	613	90.70
2-Tuples	1737	81.75	1961	83.12	1951	83.55
1-Tuples	114	68.42	290	72.41	290	73.79
Total	3097	84.31	3097	84.44	3097	84.79

Table 4.5: This table compares three kinds of single maxent models using all feature types

4.5 Chapter summary

This chapter compared interpolated and single maxent models using the standard lexical features from previous work in the field. Maxent has more flexibility, but the weights in linear interpolation are set with respect to a held-out set, possibly giving it better generalisation ability. However, we find that a single model with all possible features outperforms a set of maxent models combined using linear interpolation. A key finding of this chapter is that feature selection improves both the CB model and the maxent model, but that the cut-off points used by [Ratnaparkhi et al., 1994] and [Collins and Brooks, 1995] are too high. Feature selection is analysed in more depth in the next chapter.

Chapter 5

New Features

5.1 Introduction

Maximum entropy models have now reached accuracy rates on a par with several other machine learning approaches in this domain. This suggests that 84.5% is an upper limit given the standard features and the benchmark dataset. One escape route is to use more training data, a second is to include features that capture knowledge from other data sources. This chapter considers the second approach, leveraging the ability of maxent models to usefully combine diverse features. A key new feature type uses Latent Semantic Analysis (LSA), a method for establishing lexical preferences from a large, unlabelled dataset. The use of raw data has two key benefits:

- Less expensive to create; access to much larger training sets.
- Available for many languages. Hand-crafted, WordNet style resources have to be created for each target domain.

A model using a combination of new features achieves 85.34% accuracy, and does not require any hand-labelled (supervised) data other than the Ratnaparkhi training set.

5.2 Introducing the candidate features

The new feature types are:

- Capitalisation: $cp(x) = true$ if the first letter of x is a capital.
- Word length: $cp(x) = true$ if x has N letters.
- First n letters: $cp(x) = true$ if the first n letters of x are NNN.
- Last n letters: $cp(x) = true$ if the last n letters of x are NNN.
- LSA score. We use Latent Semantic Analysis (LSA, see below) to compare the similarity of prepositions and verbs/nouns. An example feature template is: $cp(x) = true$ if $lsa(v, p) > lsa(n1, p)$.
- WordNet senses. WordNet is a manually constructed hierarchy of word senses. We can use this to make generalised noun features such as $cp(x) = true$ if $n1$ isa "social-relation".

Some of these are explained further in the following sections.

5.2.1 Prefixes and suffixes

The first or last letters in a word potentially contain valuable information such as capitalisation, prefixes and suffixes. A simple way to capture this is to use the first or last n letters in a word as the contextual predicate. Word fragments may also function as an effective word classing scheme, since words with the same first/last letters may be used in similar ways; that is, $p(a|x) \approx p(a|fragment(x))$.

There is some degree of freedom in the way these features can be implemented. The value for n can be set globally or locally for each word type. Suffixes or prefixes could be extracted and used where applicable, essentially tuning n for each word. The best results were in fact obtained using simple global settings and minimal morphological processing: the only step is to replace digits with the token NUM.

Word fragments can be used in place of full words for any feature. Including all possible combinations lead to an explosion in the feature space, so to simplify matters only (V), (N_1), (P) and (N_2) features are included for word fragments. This policy also seems to give good results: including all possible tuples for the first/last 3 letters produces a model with 486,777 features and an accuracy of 84.60% on the validation set, but using just single-word features produces a model with 184,041 features and a still-respectable accuracy of 84.56%. Duplicating all possible tuples using word fragments leads to a very large number of redundant features and it appears that the essential information is captured in the simplest single-word tuples.

5.2.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an unsupervised method of measuring the similarity between words using co-occurrence statistics gleaned from a very large corpus. The LSA variant used here[1] starts by creating a large matrix of word co-occurrence frequencies. The rows and columns represent individual words, and each cell is number of times those two words occurred as a pair in the corpus. The next step is to reduce the dimensionality of this matrix by performing Singular Value Decomposition and keeping only n most significant dimensions. Each word is a vector in this reduced dimensionality space, and the similarity between words is essentially the distance between the vectors (actually, the distance measure used here is the cosine between the two vectors).

LSA is used as a measure of lexical preferences between a preposition and a noun or verb. If two words occur together in similar contexts, then the LSA score of that pair will be higher than for two words that rarely co-occur. If LSA is used as a classifier, then the attachment prediction is VA if $lsa(v,p) > lsa(n_1,p)$. By itself, this is a very poor classifier and only achieves 51% accuracy. Hindle & Rooth's approach achieved much higher accuracy but also used additional heuristics to train the classifier, and the $f(v,p)$ (n_1,p) frequencies were also normalised by $f(v)$ and $f(n_1)$ respectively. LSA considers word co-occurrences over the entire dataset irrespective of their syntactic

[1]We used the Colorado University LSA software available online at http://lsa.colorado.edu

role. Despite the poor performance of LSA alone, we can still use the LSA prediction for a particular word combination as a feature in the maxent model. The basic feature template is:

$$cp = \begin{cases} 1 & \text{if Attachment = A, Verb = V, Noun1 = N1, Preposition = P, LSA = A} \\ 0 & \text{otherwise} \end{cases}$$

There is also some predictive power in templates that only use a subset of the (V, N_1, P) triple. The best subsets are found to be: (V, N1, P), (V, P), (N1, P), (P). For these subsets, the final prediction is defined as the most likely prediction over all possible missing words. For example, the probability that $lsa(p) = VA$ is defined as:

$$p(lsa(p) = V) = \frac{\sum_{v,n1} f(lsa(v, n_1, p) = V)}{\sum_{v,n_1} f(v, n_1)} \tag{5.1}$$

where the sums range over all (v,n1) tuples for which we have LSA data, and f is the usual frequency function.

5.2.3 WordNet senses

WordNet has been used in many approaches to PP attachment, and we include a basic strategy here despite our prior preference for unsupervised data. As shown below, features based on LSA predictions prove to be more effective. WordNet is an English dictionary that organises words into semantic categories and hierarchies. Every defined noun belongs to one or more semantic classes (called word senses) which can be used in place of actual words in our model. Again, the hypothesis is that words belonging to the same semantic class tend to behave in similar ways. Unfortunately, English nouns often have multiple meanings and deciding exactly which sense is most appropriate in a given context is very much an open research problem (eg [Clark and Weir, 1999]). Even if the sense were known, there is still the problem of deciding which level in the sense hierarchy gives the most appropriate degree of generalisation.

For this project we adopt a rather naive approach. For each noun, take the first n senses supplied by WordNet and include all senses lying between levels i and j in

Noun	Third	Fourth
King	person	life form
oversight	inattention	basic cognitive process
taboo	opinion	belief
holidays	time period	measure

Table 5.1: Some example WordNet senses at the optimal generalisation level given our selection algorithm

the hierarchy. The parameters i, j, n are set globally. As with word fragments, we ignore combinations of words and senses, or sense tuples larger than one. The most accurate model chooses one sense per word and uses the third and fourth levels up in the hierarchy. Table 5.1 shows the word senses at these levels for some example nouns.

5.3 Comparing the candidate features

Table 5.2 shows a comparison of the various feature types, starting with the single maxent model using all possible standard lexical features described in the previous chapter. LSA is the best single new feature type, followed by prefixes and then WordNet. Finding the best *combination* of features is done semi-automatically by treating it as a parameter-setting problem and choosing the combination of features that maximises accuracy on a validation set. The best combination was found to include LSA prediction, the first three letters and the last three letters. This model has 219,536 features and achieves an accuracy of 85.34%. Table 5.3 compares the various models presented thus far in this dissertation using both the test set and ten-fold cross-validation (XVal) over the entire dataset. Assuming a normal distribution of the results, Model C does not show a statistically significant improvement in accuracy over other models although the mean accuracy is somewhat higher.

Table 5.4 shows the top fifteen features in this extended model by weight. Almost all

Method	Accuracy	Features
None	84.57	172,222
Capitalisation	84.60	+16
First Letter	84.60	+328
First 2 Letters	84.90	+1,938
First 3 Letters	84.65	+6,232
First 4 Letters	84.13	+10,023
Last Letter	84.70	+247
Last 2 Letters	84.62	+1,398
Last 3 Letters	84.65	+4,249
Last 4 Letters	84.58	+9,209
Word Length	84.55	+121
LSA Prediction	85.09	+36,833
WordNet Noun Senses	84.72	+2,310

Table 5.2: Comparing the new candidate features

Model	Test Acc.	XVal. Acc. (SD)
Model B (all feats.)	84.31	84.60 (1.08)
Model B (cut-off)	84.71	84.63 (1.12)
Model C	85.34	84.93 (0.91)
CB95	84.53	84.53 (1.15)

Table 5.3: A comparison of various models using both the test set and ten-fold cross-validation (XVal)

Position	Predicate	Category	Attachment	Weight
1	had sales of $\rightarrow$ V	lsa(v,n1,p)	V	1.2979
2	have sales of million	(v,n1,p,n2)	V	1.0324
3	have sales million	(v,n1,n2)	V	1.0324
4	reported earnings of $\rightarrow$ V	lsa(v,n1,p)	V	0.9432
5	report earnings of million	(v,n1,p,n2)	V	0.9432
6	report earnings million	(v,n1,n2)	V	0.9432
7	sold billion of $\rightarrow$ V	lsa(v,n1,p)	V	0.8822
8	yield % in YEAR	(v,n1,p,n2)	N	0.8401
9	yield % YEAR	(v,n1,n2)	N	0.8401
10	was decision of $\rightarrow$ V	lsa(v,n1,p)	V	0.8373
11	decision of $\rightarrow$ V	lsa(v,p)	V	0.8373
12	decision of NAME	(n1,p,n2)	V	0.8373
13	decision of	(n1,p)	V	0.8373
14	be decision of NAME	(v,n1,p,n2)	V	0.8373
15	be decision of	(v,n1,p)	V	0.8373

Table 5.4: The top fifteen features by weight in the extended model

of these use contextual predicates with n-tuples larger than two — as one might expect, larger contexts are usually more informative. This list also shows that the model contains many redundant features that are essentially coding for the same outcome given the same context. The following chapter deals with feature selection and removing redundant features.

5.4 Chapter summary

In this chapter we considered several additional feature types:

- Capitalisation

- Word length
- The first/last *n* letters of each word with minimal morphological processing
- The predicted attachment site using LSA
- WordNet semantic senses for both nouns

The first two feature types are derived from the words themselves, and can be seen as capturing some of the information lost during the morphological processing steps as well as clustering similar words together. The fourth feature type encodes lexical preferences gleaned from a large, unsupervised data source. The last type uses explicitly supervised additional information. In this domain, we present a model that includes the second and third categories only and achieves 85.34% accuracy. The application of large amounts of unsupervised or semisupervised data to natural language models appears to be a promising area for future research.

Chapter 6

Feature Selection

6.1 Introduction

This chapter is concerned with the problem of feature selection: finding the most useful set of features from a large pool of candidates. This process has several benefits:

- There may be spurious regularities in the data given irrelevant features, creating overfitting problems.
- Simpler models may be easier for humans to understand.
- Smaller models are more efficient to apply.

This chapter is chiefly concerned with the first point: removing features that may cause overfitting problems. As previous chapters have shown, this is often caused by low frequency features and a simply removing these prior to training is an effective solution. This chapter compares this approach with two others: Information Gain and weight variance. Freq proves to be the most robust and effective method, and operates well across a variety of models. We find that smaller models benefit most from feature elimination, but all models can be substantially reduced with little or no loss in accuracy.

6.2 Feature selection methods

6.2.1 Frequency cut-off

The policy of discarding low-frequency features has been widely used [Ratnaparkhi, 1998, Ratnaparkhi et al., 1994]. As described in Chapter 3, maxent is susceptible to overfitting with certain events, typically those with low frequency. The estimates for low-frequency events are more likely to be unreliable, but this method makes no attempt to determine whether this is the case. Nevertheless, this method has proven to be simple and effective, and it drastically reduces the model size since low frequency features (and words) are exponentially more common in natural language tasks. Figure 6.1 shows the spread of feature frequencies for Model C using log-scales on both axes. Imposing a cut-off is equivalent to drawing a horizontal line across the graph.

A frequency cut-off can either be applied globally, or different cut-offs can be applied to different feature types. The argument for the latter is that a global cut-off discriminates against more complex features. These tend to occur fewer times in the dataset and are overly represented in the feature cull (see below) yet these features are arguably more reliable. Each occurrence of a training example (a,x) is a sample drawn from the true distribution $p(A,x)$ of possible attachments for a particular context. When setting the cut-off point c, we need to decide how many (a,x) pairs are required to get a reliable estimate of $p(A,x)$ and its conditional cousin $p(A|x)$. Intuitively it seems that $p(A|x)$ has sharper peaks for more specific contexts, so fewer samples are needed to get a reliable estimate and c can be set lower. In the following experiments, we use the following general strategy:

1. Increment the cut-off for the single-word features by one (with the extra features from the previous chapter eliminated first)
2. Increment the cut-off for double features, then triples, then quadruples
3. Repeat the cycle

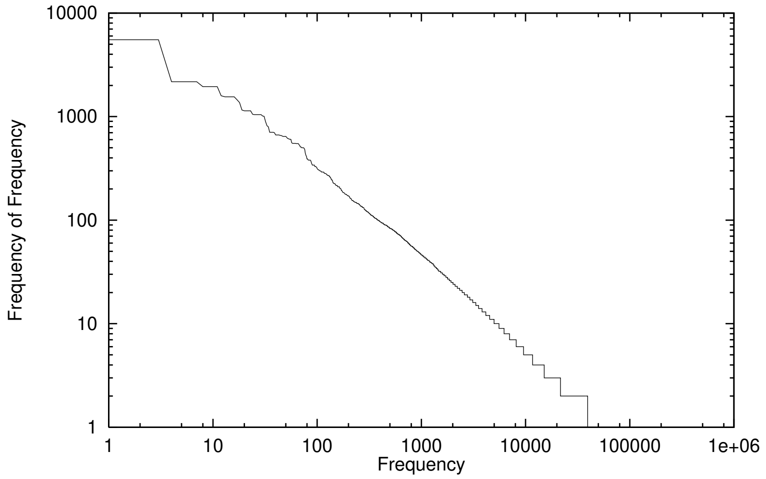


Figure 6.1: The spread of feature frequencies is a fair approximation to the power law (note the logscale on both axes)

6.2.2 Information Gain

A more theoretically motivated approach uses information theory, a useful tool for measuring entropy and uncertainty. The Information Gain (IG) measure ranks features by measuring the reduction in class entropy given that feature — actually, given a particular contextual predicate x. Recall that the entropy of a distribution distribution $p(A)$ is defined as:

$$H(A) = -\sum_{a} p(a) \log p(a) \tag{6.1}$$

If A is the attachment class of an example then $H(A)$ measures the entropy of the classes in the dataset. A predicate x effectively splits the dataset into two parts depending on whether x matches a particular example or not. The information gain of x is defined as the entropy of the overall dataset less the weighted entropy of these two parts:

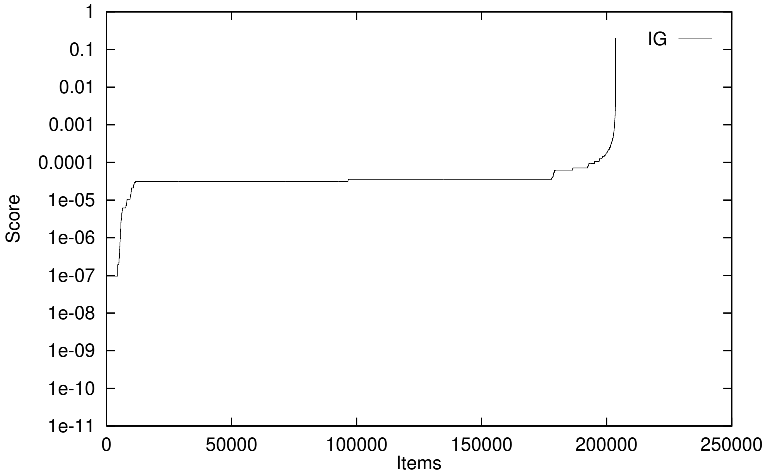


Figure 6.2: The spread of IG scores across the training set

$$IG(x) = H(A) - H(A|X) \quad (6.2)$$

$$= H(A) - (p(x)H(A|x) + p(\overline{x})H(A|\overline{x}) \quad (6.3)$$

$H(A|x)$ is the conditional entropy of A given x: it measures the amount of uncertainty in A given that we know x. If the conditional entropy is low, then we can be fairly certain about the attachment given the context. Obviously these are the features we would like to keep.

Figure 6.2 shows the spread of IG scores over the dataset. Note that many contextual predicates share the same IG score, particularly single or dual occurrences with the same attachment site. The probability of these features occurring in the training set $p(x)$ is very small so the IG score is very conservative.

6.2.3 Feature variance

This feature selection method is based on the idea that weights for features responsible for overfitting will vary widely given different training sets. Overfitting can be seen as

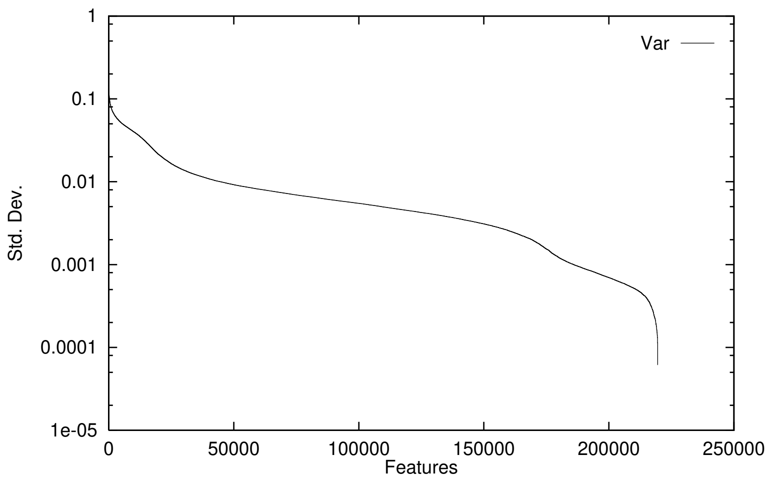


Figure 6.3: The spread of Var scores across the training set

adhering to patterns in the training data that do not exist in the real world. Features encoding for these patterns should show significant variation given a change in the training set. For this algorithm we split the training set into 10 parts and train as many models using leave-one-out cross-validation. We calculate the variance of each feature over all models (not every feature will appear in every model) and discard all features with a variance over a predefined threshold. Figure 6.3 shows the spread of variance scores over the feature set for Model C. This approach addresses at least one concern about IG, since very few features share the same score, although the main downside is the amount of computation required.

6.3 Comparing feature selection methods

The following three sections compare these selection methods on three models: the baseline model with full morphological processing (Model A), the single maxent model from Chapter 4 (Model B) and the model using the additional features from Chapter 5 (Model C). These models have progressively more redundant features, and we find that Model A benefits the most from feature selection, and Model C shows little or no benefit. However, the feature sets for all models can be drastically reduced with little or no loss in accuracy.

6.3.1 Model A

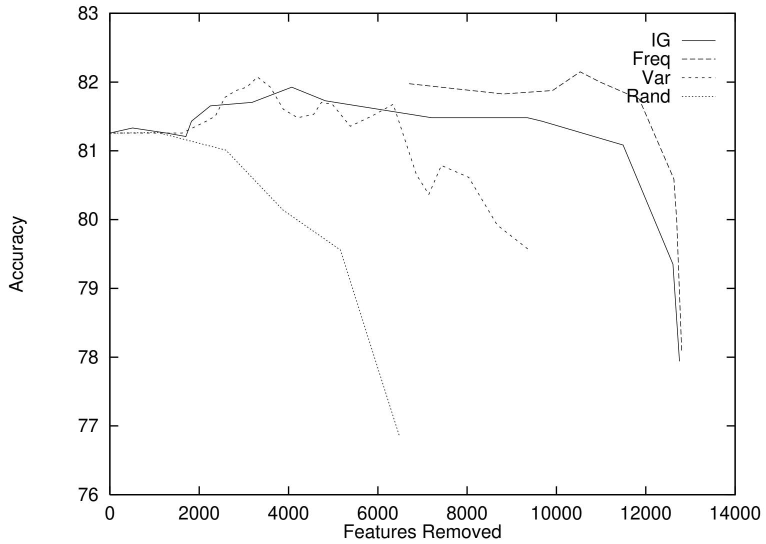
Figure 6.3.1 shows the results of training this baseline model with different feature sets obtained using different feature selection methods. For comparison purposes we also show a *Rand* method that simply removes features at random (the accuracy results for this method are averaged over 20 runs).

The final results are summarised in the table below. The settings were chosen to maximise accuracy on the validation set. All (sensible) selection methods produce models with fewer features that outperform the full model. The Freq method produces the smallest model which also has the highest accuracy. The Var method is highly erratic but the optimal value on the validation set also produces a respectable result against the test set.

Method	Setting	Features	Accuracy
None		12,972	82.08
Freq	$c = 4$	2,441	82.47
IG	$g = 0.00002$	8,889	82.44
Var	$v = 0.17$	9,761	82.11

6.3.2 Model B

We now compare the feature selection methods on the single maxent model presented in the last chapter. There are 172,222 candidate features and the full model has an accuracy rate of 84.31%. The results of the slimmed-down models are shown in Figure 6.4, and the table below shows the optimal settings for each method. The frequency cut-offs used are $c = 0$ for quadruples and $c = 1$ for triples, doubles and singles. The Var method shows rather disappointing performance in this case; Section 6.4 discusses some possible reasons for this.



Method	Setting	Features	Accuracy
None		172,222	84.31
Freq	$c = 0,1,1,1$	25,777	84.79
IG	$g = 0.00006$	18,934	84.55
Var	$v = 0.4$	162,679	84.65

6.3.3 Model C

The results of feature selection in this model are shown in Figure 6.5. Rand performs surprisingly well here, illustrating the high degree of redundancy in the feature set. The table below summarises the results. Once again, Var is the worst performer, underperforming even Rand. There are two obvious maxima in the IG curve and two corresponding entries in the table below. IG and Freq show similar accuracy levels; in this case removing a small number of features using IG produces the best results.

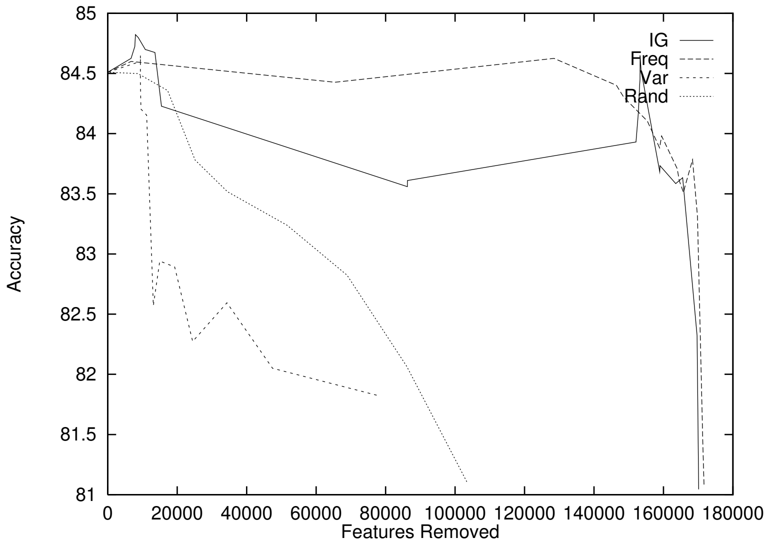


Figure 6.4: Comparing the different feature selection methods on Model B

Method	Setting	Features	Accuracy
None		219,536	85.34
Freq	$c = 1$	39,443	84.99
IG	$g = 0.000001$	210,610	85.47
IG	$g = 0.00005$	29,772	84.60
Var	$v = 0.08$	218,647	84.71

6.4 Discussion

These experiments show that both IG and Freq are reasonable feature selection methods, and this immediately raises the question about which features each is removing. This section shows that although IG is inherently less biased towards complex features, Freq is the best method for eliminating features that may potentially be causing overfitting. Figure 6.8 is a scatterplot showing the correlation between frequency of contextual predicates (not features) and their IG scores. The lack of correlation here shows that each method eliminates quite different features, although IG does tend to

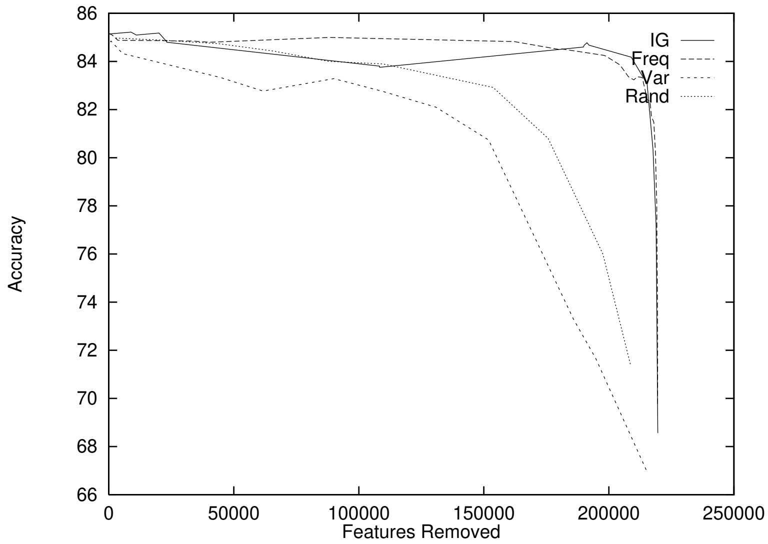


Figure 6.5: Comparing the different feature selection methods on Model C

assign higher values to more frequently occurring features. This suggests that the two methods can be usefully combined, and indeed applying a frequency $c = 1$ *and* and IG cut-off $g = 0.0003$ produces a model with 28,508 features (some 10,000 fewer than when using Freq alone) and 85.02% accuracy. The combination of these two methods deserves to be explored in other domains.

Figure 6.6 is a scatterplot of feature weights and feature frequencies. The weights clustered around zero in the middle of the graph are of minimal informative value, but the weights with extreme values are believed to be overfitting the training data. Freq clearly eliminates these high weight features first; however, many other low-weight features are also eliminated at this first chop. Figure 6.7 shows the same data for IG contexts and their corresponding weights. Low IG scores are removed first, so weights are removed from left to right. In contrast to Freq, IG does not first eliminate the highly weighted, possibly overfitting features.

Table 6.4 shows the proportion of different features removed by each method. Both are biased towards removing more complex features, but in the case of IG this bias is less pronounced. To compensate for this bias, the Freq method can be tuned for particular

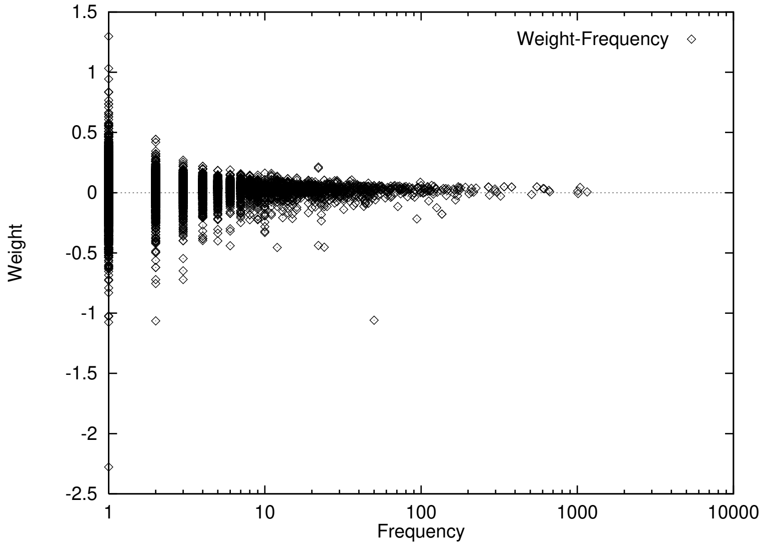


Figure 6.6: A scatterplot of feature frequency against feature weight after training for Model C

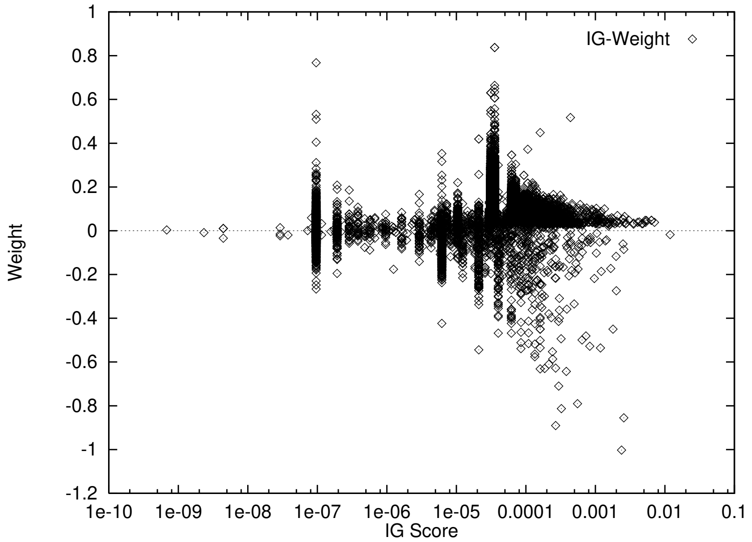


Figure 6.7: A scatterplot of IG score against one feature weight after training for Model C

Feature	Freq	IG
(V, N_1, P, N_2)	10.0	9.5
(V, N_1, P)	8.0	7.6
(V, N_1)	6.9	6.9
(V, P)	2.1	2.3
(N_1)	1.5	2.0
(V)	0.5	0.7

Table 6.1: The proportion of various feature types removed by the Freq and IG selection methods

feature types. There are currently seven different types of features in our model: four *n*-tuple features, the first and last 3 letters, and the LSA prediction. Of course, there are actually four different 3-tuple feature types, 6 different 2-tuple feature types and so on: 29 in all, each of which can potentially be tuned separately. Attempting to hand-tune these is a formidable task, and treating them as free parameters to be tuned against a validation set usually leads to overfitting problems.

The Var method shows rather disappointing performance, significantly underperforming Rand in the larger models. The minimal correlation between variance and frequency is shown in Figure 6.9. High frequency features have average variance and the scores vary less (there is less variance in the variance). The opposite is true for low-frequency scores. One problem appears to be the sensitivity of the variance measure to outlying values. Using variance penalises features that have one terrible run but otherwise proved to be generally reliable predictors. Indeed, selecting features that have a *high* variance is a better feature selection method, although still only outperforming Rand by a small margin. A possible direction for future research is to use a less sensitive measure of variation such as interquartile range. However, this method is also by far the most expensive to perform, requiring several complete training runs.

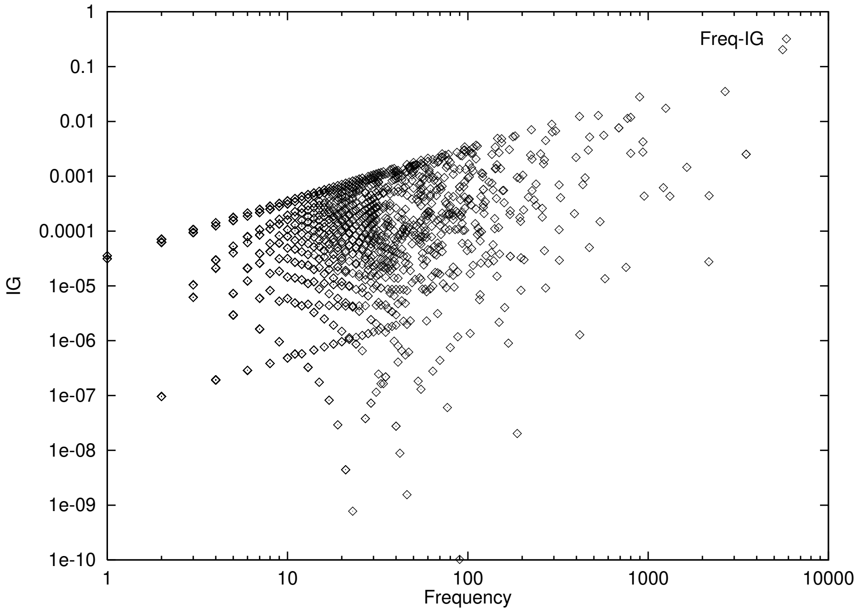


Figure 6.8: The correlation between IG and Freq scores

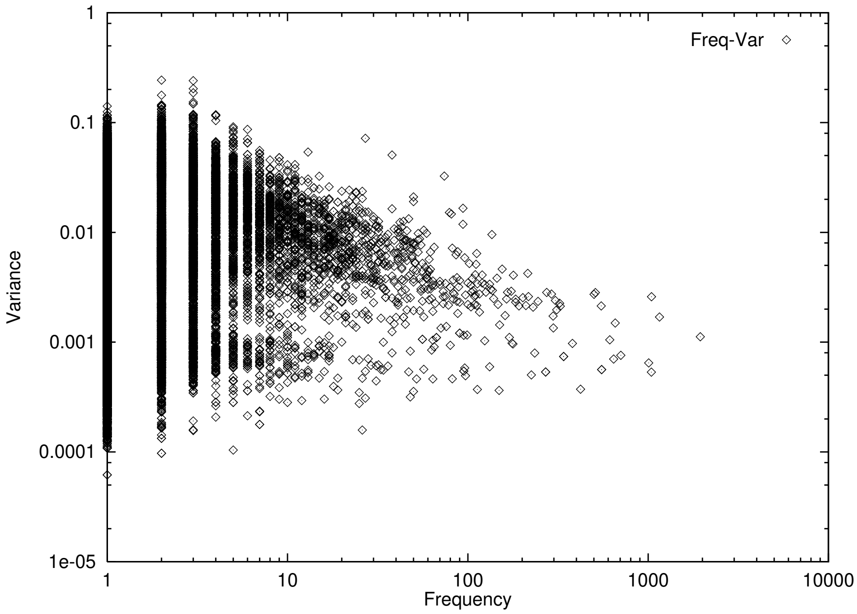


Figure 6.9: The correlation between Freq and Var scores

6.5 Chapter summary

Of the three frequency selection methods presented in this chapter, Freq the cheapest to implement and is robust across a range of models. It also best fulfills the desire to remove features with extreme weights that are possibly caused by overfitting. Information Gain is a promising technique, and in some cases it is superior to the widely-used Freq technique. It has a better theoretical foundation and is less biased towards removing complex features without having to tune any parameters. Eliminating features with very small IG values often leads to the largest accuracy gains if not the smallest models. An interesting avenue for future work is to compare and combine the two methods over different domains.

Chapter 7

Ensembles

7.1 Introduction

There is a considerable body of work in machine learning showing that the combined prediction of a collection of models is often more accurate and generalises better than the prediction of a single model [Dietterich, 2000]. This chapter explores ensembles of maxent models for this domain. We use two methods that manipulate the training set: bagging [Breiman, 1996] and bagging with additional random noise. A noisy bagging ensemble of Model C classifiers achieves a mean accuracy rate of 85.53$\pm$0.15%

7.2 What makes a good ensemble

Ensemble methods are not guaranteed to improve performance. For example, if we have ten classifiers making exactly the same predictions, then clearly their combined prediction will not be useful. Indeed, Tumer & Ghosh (1996) have shown that when averaging an ensemble of unbiased but correlated classifiers, the error of an ensemble is directly related to the correlation of errors between between classifiers.

This section presents the tools used in this dissertation to measure diversity and understand why some ensemble methods are more effective than others. Before we delve into this, note that there are three outputs we are interested in:

- The estimate of the posterior probability $\tilde{p}_i(a|x)$
- The predicted class label $c_i(x) = \text{argmax}_a\ \tilde{p}_i(a|x)$
- The true class label, as given by an oracle classifier $d(x)$.

The degree of correlation between classifiers is essentially the probability that two c_i and c_j classifiers make different errors:

$$p_{error} = \sum_x p(c_i(x) = a, c_j(x) \neq a) \tag{7.1}$$

p_{error} can be averaged over all classifier pairs to obtain a measure of error correlation, but not one whose results are particularly intuitive.

A common measure of correlation is Pearson's correlation coefficient: the normalised covariance of two variables ϕ_i and ϕ_j:

$$r = \frac{covar(\phi_i, \phi_j)}{sd(\phi_i)sd(\phi_j)} \tag{7.2}$$

The variable ϕ of interest here is a binary variable indicating whether the classifier c_i made an error on example x:

$$\phi_i(x) = \begin{cases} 1 & \text{if } d(x) = a, c_i(x) \neq (x) \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

The expected value of ϕ_i over the test set is simply the error rate of c_i. This correlation measure has a natural meaning of 0 if the two variables are statistically independent, and 1 or -1 if they are perfectly (negatively) correlated. The error correlation of an ensemble is defined as the average error correlation between all pairs.

An ideal ensemble of classifiers makes de-correlated or negatively correlated errors without sacrificing individual accuracy. Unfortunately, the maxent ensemble methods considered here hinder the classifier in some way, so there is a trade-off between diversity and accuracy. The latter is measured in the obvious way: by calculating the mean accuracy of the individual classifiers.

The predictions of each classifier must also be combined in some way. One simple combination method is majority voting, where the final prediction is simply the class chosen by the most individual classifiers. This only considers the predicted class labels of each classifier, not the posterior probability. Two other popular schemes that do consider the probability estimate are summing:

$$\tilde{p}_e(a|x) = \frac{1}{N} \sum_{i}^{N} \tilde{p}_i(a|x) \tag{7.4}$$

and multiplying:

$$\tilde{p}_e(a|x) = \frac{1}{Z} \prod_{i}^{N} \tilde{p}_i(a|x) \tag{7.5}$$

where $\tilde{p}_e(a|x)$ is the final ensemble estimate, $\tilde{p}_i$ is an individual classifier and Z is a normalising constant in (7.5). Note that the multiplication rule is equivalent to a sum of log probabilities. We find that the sum rule produces the most accurate ensembles given the ensemble methods used in this dissertation. Majority voting may be more appropriate if the predictions of the individual classifiers are unreliable or not indicative of their confidence in their decision, but as we show in Section 8.4, the predictions of maxent classifiers in this domain are trustworthy.

7.3 Creating diverse ensembles of maxent models

There are many ways in which a dataset or a learning algorithm can be manipulated to produce model ensembles [Dietterich, 2000]. We concentrate here on methods that manipulate the training data: bagging and noisy bagging.

7.3.1 Bagging

Each classifier in a bootstrap aggregation (bagging) ensemble is trained on a subset, called a bootstrap replicate, of the training data [Breiman, 1996]. Each replicate is created by randomly sampling m items from the original dataset of n items, a way of distorting the training set that can be seen as enhancing any intrinsic noise. Breiman (1996) suggested that the effectiveness of bagging depends crucially on the instability of the learning algorithm. Unstable learners such as decision trees and neural networks are capable of producing very different classifiers given small changes in the training data. Bagging can harm the accuracy of stable learners because each classifier sees less data: for a large m, each replicate contains on average only 63% of the original dataset. The experiments below show that this is the case with maxent: the individual classifiers are more inaccurate than classifiers trained on the full training set, but diversity in the ensemble just compensates for this.

7.3.2 Noisy bagging

This bagging variant creates more diverse ensembles by adding uniform, unbiased noise to each aggregate. For this domain, this means that each example has its attachment site flipped with some (small) probability. Noisy training sets have been shown to improve the generalisation ability of neural networks [Sietsma and Dow, 1991, Raviv and Intrator, 1996] showed that noisy bagging allowed the use of aggregates much larger than the original training set without producing highly correlated classifiers. Noisy bagging is a more effective ensemble method for maxent classifiers.

7.4 Experimental results

Table 7.1 compares ensembles of various models using simple bagging. “Single Acc.” refers to the accuracy of a single maxent model trained using the normal training set, “Ind. Acc.” is the mean accuracy of the individual classifiers and “Ens. Acc.” is

Model	Features	Single Acc.	Ind. Acc	Correl.	Ens. Acc.
Model A	12,972	82.08	81.79	0.71	82.30
Model A ($c = 1$)	6,272	82.63	81.48	0.67	82.56
Model B	172,222	84.31	83.72	0.69	84.63
Model B ($c = 1$)	25,777	84.50	83.39	0.67	84.82
Model C	219,536	85.34	83.80	0.71	84.70
Model C ($c = 1$)	39,443	84.99	83.66	0.70	84.90

Table 7.1: Comparing bagging with different constituent classifiers. Ensembles of larger models make for better classifiers, and using larger aggregates lifts the accuracy rate of Model C.

the mean accuracy of the ensemble. Each classifier was trained on an aggregate of 20,801 examples, the same size as the training set, with 25 classifiers per ensemble. The results shown are the mean accuracy figures for 15 ensembles created randomly from a pool of 100 classifiers.

Models B and C best fit the criteria for diverse but accurate ensembles. Once again, classifiers using a frequency cut-off perform better than those without. It is possible to alter the selection strategy for each aggregate to ensure that all features occur more than once; however a simple frequency cut-off is found to work better in practice. The remainder of these experiments use the best two models from these pilot experiments: Models B and C with cut-offs $c = 1$.

7.4.1 Bagging

The key parameter for maxent bagging in this domain is aggregate size. Figures 7.1 and 7.2 show the results of using 51 Model B and Model C aggregates for various aggregate sizes. The classifiers in each ensemble are drawn randomly from a pool of 200 classifiers. The mean accuracy of a Model B ensemble using 30,000 items is 85.10%, an absolute increase of 0.31% over the single model. The mean accuracy for

	Model B			Model C		
Agg. (k)	Ind. Acc.	Error Corr.	Ens. Acc.	Ind. Acc.	Error Corr.	Ens. Acc
20	83.39	0.68	84.63	83.67	0.70	84.73
30	84.03	0.72	84.89	84.90	0.78	84.90
40	84.09	0.76	84.72	85.13	0.84	85.13
50	84.15	0.84	84.53	85.18	0.88	85.23

Table 7.2: Comparing different aggregate sizes for Models B and C using bagging

51 Model C classifiers using 50,000 items is 85.23±0.7%, an absolute *decrease* of 0.11%.

Table 7.2 shows the accuracy and correlation coefficients for both models and various aggregate sizes. Increasing the ensemble size produces diminishing returns since it gets harder and harder to create uncorrelated classifiers, although the individual classifiers get more and more accurate due to the additional training data. Larger aggregates contain more of the many unique singleton features in the training set. For example, an aggregate with 20,000 items has on average 147,000 unique Model C features, but a 50,000-item aggregate has on average 203,000 unique features.

7.4.2 Noisy Bagging

The key parameters in this method are the noise rate and the aggregate size. Table 7.3 shows the effect of varying the noise rate using 30,000-item aggregates. Again, these figures are mean results for 15 ensembles of 51 classifiers drawn randomly from a pool of 150 classifiers. As the noise level increases, the individual accuracy decreases monotonically but the error correlation also falls. A noise rate of 10% is an optimal trade-off between the two. Contrary to findings in [Raviv and Intrator, 1996], we find that larger aggregate sizes are of little benefit in this domain, as Figures 7.3 and 7.4 show. The independent variable here is the aggregate size and the noise is constant at 10%. A Model C ensemble achieves a mean accuracy of 85.53±0.15% given 10%

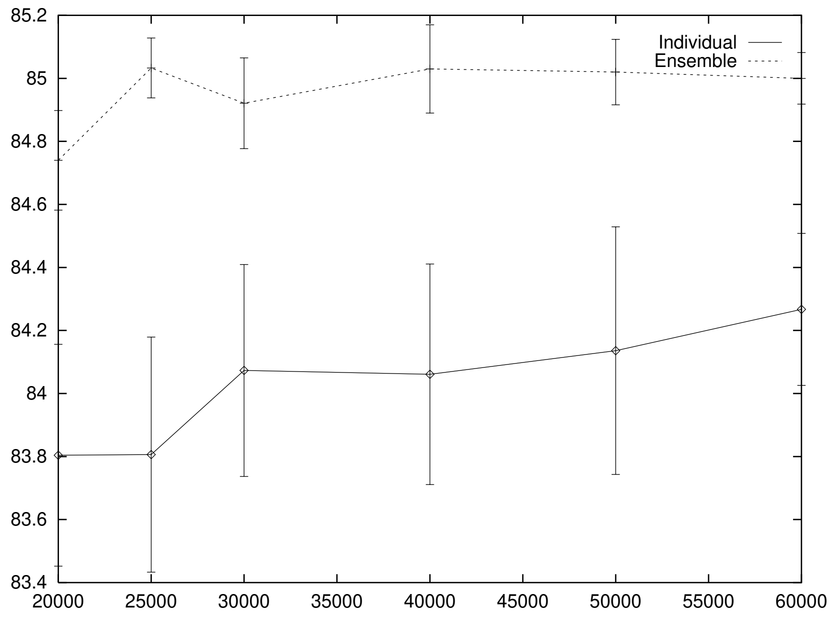


Figure 7.1: Results of bagging with Model B for varying aggregate sizes

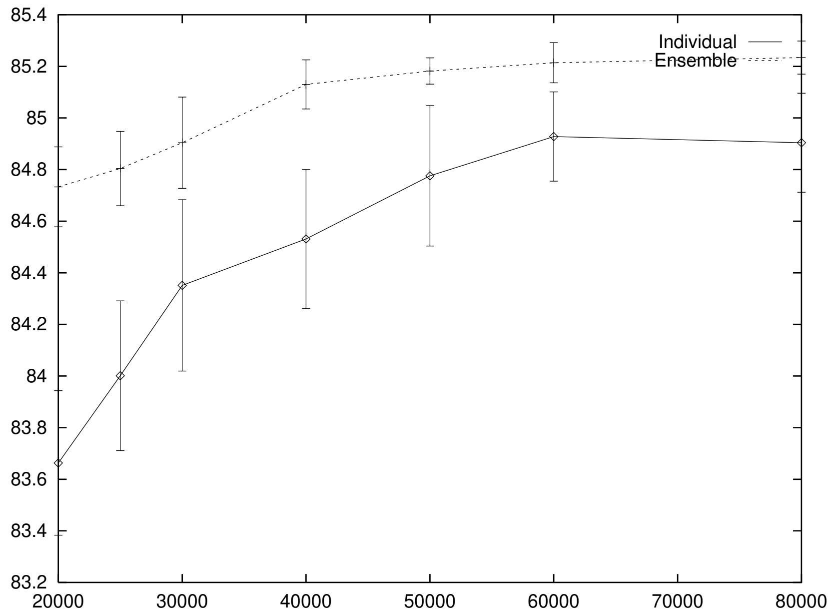


Figure 7.2: Results of bagging with Model C for varying aggregate sizes

	Model B			Model C		
Noise	Ind. Acc.	Error Corr.	Ens. Acc.	Ind. Acc.	Error Corr.	Ens. Acc
5%	83.58	0.60	85.22	83.88	0.66	85.27
mcbain 10%	81.38	0.50	85.20	83.06	0.59	85.53
15%	79.40	0.40	84.93	81.68	0.46	85.18
moe 20%	76.99	0.31	84.20	79.85	0.42	84.95
30%	70.46	0.16	83.94	74.37	0.23	84.10

Table 7.3: Comparing different noise rates for noisy bagging, aggregate size is 30,000

noise and 30,000-item aggregates.

7.5 Discussion

Ensemble methods only produce marginal improvements over comparable single model classifiers. The problem may lie with the ensemble methods used here and the use of maxent as the underlying classifier. Neither bagging nor noisy bagging produced the desired decorrelated or negatively correlated classifiers, implying that maxent models are stable learners and bagged maxent ensembles are of only a small benefit. Another possibility is that we are reaching the upper limit of what is possible using lexical features alone, since these accuracy rates are now approaching the human accuracy level of 88.1%.

It is possible that better results could be obtained using other ensemble methods. There is evidence [Osborne, 2001] that modifying the initial weights produces a diverse set of classifiers. Another approach might be to use decorrelation measures explicitly when creating ensembles. This has been done with neural nets by attaching a correlation penalty adjustment to each weight [Rosen, 1996]. Log-linear models can be trained to maximise the similarity to any distribution, not just the maximum entropy one. This initial distribution could be altered separately for each classifier to produce more ef-

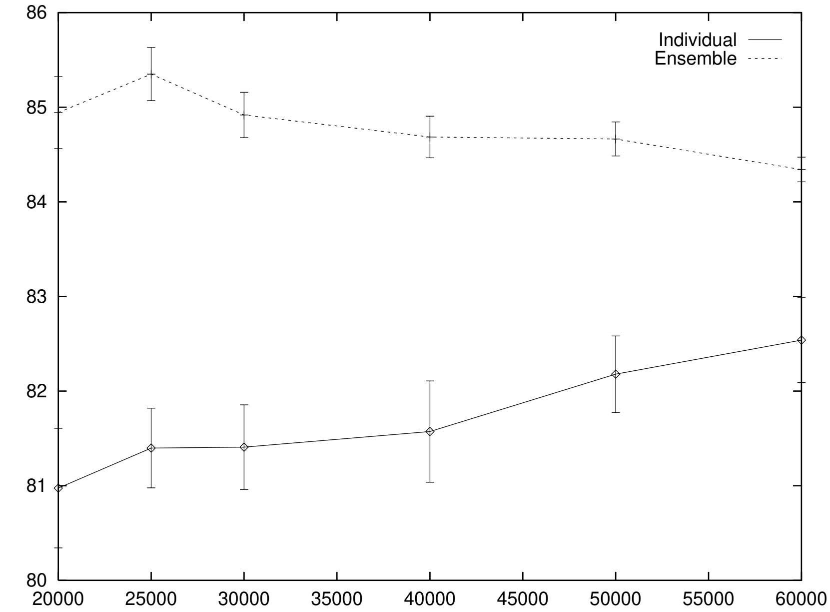


Figure 7.3: Results of using noisy training sets with Model B for various aggregate sizes. The noise level is constant at 10%

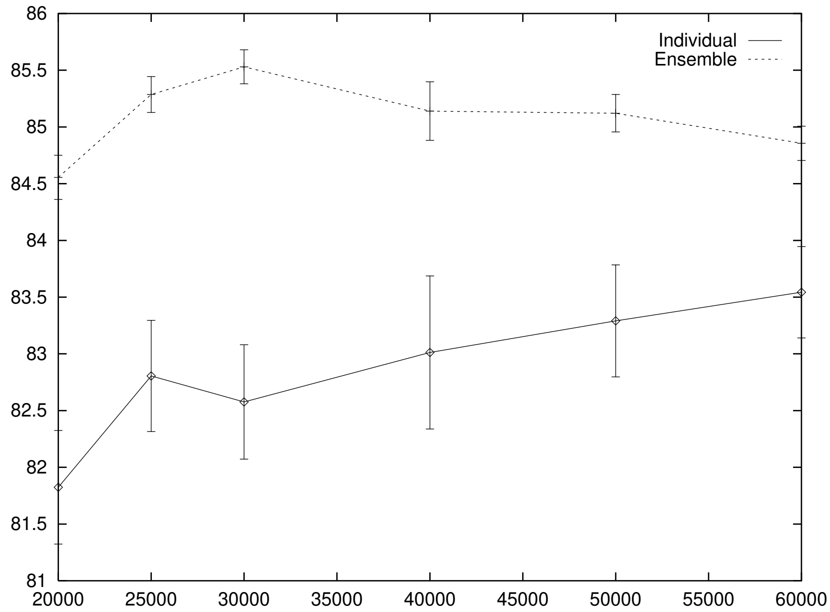


Figure 7.4: Results of using noisy aggregates with Model C for various aggregate sizes. The noise level is constant at 10%

fective ensembles. A related idea is found in [Beeferman et al., 1997] where a penalty function is added to a maxent language model.

7.6 Chapter summary

This chapter presented two ensemble methods for maxent models. Simple bagging is found to be less suitable owing to the stability of the individual maxent classifiers. Noisy bagging injects further randomness into the training procedure and produces more decorrelated classifiers. We achieve $85.53 \pm 0.15\%$ accuracy with an ensemble of 51 maxent classifiers created using noisy bagging, marginally above the accuracy rate for the comparable single model. Further improvements may be gained using better ensemble methods, but these results may well be approaching an upper bound on the task given lexical features.

Chapter 8

Analysis and Conclusion

8.1 Introduction

This chapter is divided into three parts. Firstly we compare the maxent results with classifiers constructed using the *Naive Bayes* approach, a comparable probabilistic technique that makes additional assumptions about the data. In this domain, Naive Bayes classifiers are less accurate than equivalent maxent classifiers, justifying the use of the more powerful maxent framework.

Secondly, the output of a particular maxent model, Model B, is analysed to gauge the effect of ambiguity, noise and sparse data. This analysis also suggests some areas for future research: we find for example that the lack of training data is much more harmful than ambiguity. Using this analysis we also present the results of using a post-hoc *reject* option where we refrain from classifying examples where the classifier had a low confidence. A maxent classifier is shown to be less confident about ambiguous cases, meaning that we can eliminate low-confidence items to improve precision. Finally, the last section summarises the key points of this dissertation and points out additional directions for future research.

8.2 Comparison with Naive Bayes

A key strength of maximum entropy models is the ability to combine large numbers of features in a single model. This section compares maxent to the *Naive Bayes* (NB) framework, a simpler but potentially less powerful way of doing to the same trick. The two approaches differ chiefly in their assumptions about how features interact. Features are often statistically dependent — the value of one feature is not independent of other feature values. For example, the value of a triple (v, p, n_1) is highly dependent on the values of the constituent doubles (v, p) and (p, n_1). Maxent can model these dependencies, but we pay a price in the form of an iterative procedure to find the maximum likelihood model. The *naive Bayes* framework takes the opposite approach: it finds the maximum likelihood model analytically, but at the cost of assuming that all features are independent of one another. This can lead to flawed models if these assumptions are violated, although they are suprisingly robust in the face of such violations. Naive Bayes classifiers are quick to train, efficient to run and have been shown to be effective in several domains, in some cases outperforming maxent [Nigam et al., 1999].

Starting with the usual conditional distribution $p(A|X)$, we can break down the context X into a set of features[1] $X_1, X_2, X_3...$ to give the posterior probability of an attachment $p(A|X_1, X_2, X_3, ..)$. Using Bayes' rule this becomes:

$$p(A|X_1, X_2, X_3, ...) = \frac{p(X_1, X_2, X_3, ...|A)p(A)}{p(X_1, X_2, X_3, ...)} \tag{8.1}$$

The denominator is a normalising factor that can be ignored when comparing different class probabilities. The likelihood term $p(X_1, X_2, X_3, ...|A)$ can be factored using the chain rule to give:

$$p(X_1, X_2, X_3, ...|A) = p(A)p(X_1|A)p(X_2|X_1, A)p(X_3|X_2, X_1, A)... \tag{8.2}$$

A number of parameters in this distribution rises exponentially with the number of

[1] In this case, we using the usual machine learning definition of a feature being attribute of an example and not a maxent feature function.

Features	NB Accuracy	ME Accuracy
Model A	81.63	82.50
Model B	83.66	84.47
Model C	84.40	85.32

Table 8.1: Results for Naive Bayes given various feature sets

features and so estimating (8.2) is usually an infeasible task for any sizeable problem. But if we make the rather large assumption that all features are independent of one another, then the condition feature variables disappear and we have:

$$p(C|X_1, X_2, X_3, ...) \approx p(X_1|C)P(X_2|C)P(X_3|C)...P(C) \tag{8.3}$$

Each $p(X_n|C)$ term can be efficiently estimated from the training data using maximum likelihood and the number of parameters now scales linearly with the number of features. Each term is usually smoothed to avoid the usual overfitting problems; in this case we use basic add-n (Laplacian) smoothing (here $n = 0.5$).

Table 8.1 compares the accuracy of the three favourite maxent classifiers with the equivalent NB classifier. The usual morphological processing was applied to the dataset in all cases and the maxent models had a global frequency cut-off $c = 1$. As with the maxent models, larger NB models produced more accurate results. The best NB model also uses the highly redundant feature types described in Chapter 5, even though the feature set violates the underlying independence assumptions to a surprising degree. However in each case, the maxent model is significantly more accurate than its Naive Bayes counterpart.

Type	Number	% of category
Locative	27	12.5
Benefactive	36	16.6
Idioms	12	5.5
Context	92	42.4
Nonsensical	50	23.0
Ambiguous	**217**	**34.8**
Unambiguous & $<$ 3 words	264	65.0
Unambiguous & $>=$ 3 words	142	35.0
Unambiguous	**406**	**65.2**
Total	**623**	**100.0**

Table 8.2: An analysis of the errors made by Model B on the validation set

8.3 Error analysis and the problem of unknown words

It is important to identify where our model might be going wrong. As a starting point, we take the Model B with no frequency cut-off. This classifier makes 623 errors on the development test set (84.58% accuracy). Table 8.2 divides these errors into ambiguous cases (using the ambiguity classes described in the Introduction) and unambiguous cases. The unambiguous cases are further split depending on whether the most complex applicable feature used three or more context words. The *context* category covers examples where the specified attachment site appeared to be wrong on the evidence of the (v, n_1, p, n_2) words alone. It is difficult to judge whether this is due to contextual ambiguity or whether it could be resolved with access to just the surrounding sentence; it may also include cases of annotator error in the WSJ. The *nonsensical* category covers cases that simply do not make sense and there is no apparent reason to prefer any particular attachment site over another.

The largest single source of ambiguity is in the *context* category. The accuracy of human experts rose 5% given access to the entire sentence; we believe that classifiers will also benefit from more context in the same sentence. The main obstacle is the sparse data problem, and it remains to be seen how very large maxent models with larger contexts perform on this problem. Aside from extraction error, all other sources of ambiguity are relatively minor. This is an encouraging result, suggesting that the vast majority of PP's are of the relatively benign type.

In the unambiguous cases, the majority of erroneous cases use very little context (two context words or less). Some of these cases must be attributed to classifier bias, but an obvious area for future research given this results is simply to use much more training data. This will require the use of less supervised datasets, driving up the extraction error rate. The trade-off between these two areas deserves further research.

8.4 Using probabilities as a rejection criterion

This section examines whether maxent can be used to identify the ambiguous cases. An advantage of using a probabilistic model such as maxent is that the output can be interpreted as a degree of confidence in the result. If the difference between the probabilities $P(NA|x)$ and $P(VA|x)$ is small, then the attachment sites are almost equally likely and the maxent model is not particularly confident about its prediction. Hindle & Rooth (1993) used a similar log-ratio between the two probabilities to gauge the confidence of their classifier.

A new classifier is created that uses the maxent predictions but rejects any examples where the difference is below a certain threshold. Table 8.3 shows the effect of this *reject* option. At the 10% confidence level, this new classifier still accepts almost 95% of the examples with 86.2% accuracy, and almost three-quarters of the rejected examples are indeed ambiguous. This is a gratifying result, considering that ambiguous examples only make up a third of all errors.

Margin	Accuracy	% Rejected	Ambig. Rejects
0.01	84.7	0.6	77.8
0.02	84.7	1.1	71.9
0.05	85.3	3.2	74.3
0.1	86.2	6.5	72.3
0.2	88.2	12.7	69.5
0.3	90.3	19.3	67.1
0.4	92.1	25.5	65.6

Table 8.3: An comparison of classifiers that reject low-confidence examples

8.5 Conclusion

The first goal of this dissertation was to show that maxent classifiers compare favourably to other state-of-the-art algorithms using lexical features, for example Collins & Brooks back-off model, Memory-Based Learning and Boosting. Chapter 4 presented a classifier that improves on previous maxent results by using additional feature tuples and imposing lower frequency cut-offs. These results also slightly exceed those of other machine learning approaches using no additional supervision. We have also shown that a classifier using a single maxent model performs comparably to a backed-off classifier using a series of maxent models. A backed-off model has the advantage of having additional parameters tuned against a validation set, whereas the single model maximises the likelihood of the training data. However, the single model makes better use of the features and interactions between features.

Maxent models are able to combine diverse information sources and in this dissertation we were able to derived lexical features and LSA predictions to boost accuracy to 85.32%. LSA captures lexical preferences from unsupervised data source, giving access to very large datasets without the need for human supervision. The other useful feature category uses suffixes and prefixes, capturing some of the information lost by applying the morphological processing steps (Section 4.3.1).

Chapter 6 compared three feature selection methods: frequency cut-offs, information gain and weight variance. The main goal was to eliminate features that are uninformative but also may cause maxent to overfit the training data. A simple frequency cut-off best achieves this goal, and it also reliably produces the most accurate cut-down models over a range of different model sizes. The information gain measure removes quite different features than Freq and the two techniques can be usefully combined.

The final section of this dissertation examined ensembles of maxent classifiers creating using bagging. This is not a particularly appropriate technique for stable maxent classifiers since each individual classifier suffers from the lack of training data and the final ensembles are not overly diverse. Improved diversity can be achieved by adding uniform noise to each aggregate, and a noisy bagging ensemble outperforms an equivalent single classifier with an accuracy of 85.53%. This may be near an upper limit given the dataset and lexical features.

Much of the previous work in this field has consisted of applying different machine learning techniques to the domain. However, we believe that the prepositional phrase attachment task has largely been solved as a data-modelling problem: given the standard Ratnaparkhi dataset and the usual lexical features, various learning algorithms have now reached an apparent upper bound on the problem of around 84.5%. We believe that the most significant gains in this problem will come from two areas:

- Using information from the surrounding context and additional data sources. LSA shows some promise, and being an unsupervised technique it can be used to gather large datasets and applied to languages without WordNet-style resources.
- The use of larger, possibly noisy, training data. Pantel & Lin (2000) used an automatic parser to generate very large amounts of presumably quite noisy training data. However, we believe that the classifier used in that report is not optimal, and a worthwhile extension to this project would be to use larger amounts of training data with the maxent algorithm.

Maxent is well-placed to take advantage of the first development, since it is an ideal framework for combining different information sources. Ratnaparkhi (1998) experimented with the second idea for PP attachment with limited success, but this idea deserves additional research using better extraction methods and comparing results in different domains.

Bibliography

[Altman and Steedman, 1988] Altman, G. and Steedman, M. (1988). Interaction with context during human sentence processing. *Cognition*, 30:191–238.

[Beeferman et al., 1997] Beeferman, D., Berger, A., and Lafferty, J. (1997). A model of lexical attraction and repulsion. In Cohen, P. R. and Wahlster, W., editors, *Proceedings of the 35th Meeting of the ACL and 8th Meeting of the EACL*, pages 373–380. Association for Computational Linguistics.

[Berger, 1997] Berger, A. (1997). The improved iterative scaling algorithm: A gentle introduction. http://www.cs.cmu.edu/afs/cs/user/aberger/www/ps/scaling.ps.

[Berger et al., 1996] Berger, A., DellaPietra, S., and DellaPietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[Brown et al., 1992] Brown, P. F., DellaPietra, V. J., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):468–479.

[Charniak, 1999] Charniak, E. (1999). A maximum-entropy-inspired parser. Technical Report CS-99-12, Brown University, Providence, RI.

[Chen and Chen, 1996] Chen, K. and Chen, H. (1996). A rule-based and mt-oriented approach to prepositional phrase attachment.

[Clark and Weir, 1999] Clark, S. and Weir, D. (1999). An iterative approach to estimating frequencies over a semantic hierarchy. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 258–265.

[Collins and Brooks, 1995] Collins, M. and Brooks, J. (1995). Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on*

Very Large Corpora, pages 27–38, Somerset, NJ. Association for Computational Linguistics.

[Darroch and Ratcliff, 1972] Darroch, J. and Ratcliff, D. (1972). Generalised iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480.

[DellaPietra et al., 1997] DellaPietra, S., DellaPietra, V., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

[Dietterich, 1998] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems, First International Workshop*, pages 1–15, Cagliari, Italy. Springer Verlag.

[Ford et al., 1982] Ford, M., Bresnan, J., and Kaplan, R. (1982). *The Mental Representation of Grammatical Relations*, chapter A competence based theory of syntactic closure. MIT Press, Cambridge, MA.

[Franz, 1996] Franz, A. (1996). *Automatic Ambiguity Resolution in Natural Language Processing: An Empirical Approach*, volume 1171 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin.

[Frazier, 1979] Frazier, L. (1979). *On comprehending sentences: Synactic parsing strategies*. PhD thesis, University of Connecticut, West Bend, IN.

[Gibson, 1991] Gibson, E. (1991). *A computational Theory of Human Linguistic Processing: Memory Limitation and Processing Breakdown*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

[Gibson and Pearlmutter, 1994] Gibson, T. and Pearlmutter, N. (1994). *Perspectives on Sentence Processing*, chapter A corpus-based analysis of psycholinguistic constraints on PP attachment. Lawrence Erlbaum Associates.

[Hindle and Rooth, 1993] Hindle, D. and Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120.

[Jaynes, 1957] Jaynes, E. (1957). Information theory and statistical mechanics part II. *Physical Review*, 108.

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

[Kimball, 1973] Kimball, J. (1973). Seven principles of surface structure parsing in natural language. *Cognition*, 2.

[Lau, 1994] Lau, R. (1994). Adaptive statistical language modelling. Master's thesis, MIT, Cambridge, MA.

[Mason et al., 2000] Mason, L., Baxter, J., Bartlett, P., and Frean, M. (2000). Boosting algorithms as gradient descent in function space. *Neural Information Processing Systems*, 12:512–518.

[Miller, 1990] Miller, G. (1990). Wordnet: An on-line lexical database. *International Journal of Lexicography*, 4(3).

[Nigam et al., 1999] Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67.

[Osborne, 2001] Osborne, M. (2001). Just how good is maximum entropy? an empirical investigation using ensembles of memd models for attribute-value grammars. Draft paper at http://www.cogsci.ed.ac.uk/ osborne/memd.ps.

[Ratnaparkhi, 1996] Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the Empirical Methods in Natural Language Processing*, University of Pennsylvania.

[Ratnaparkhi, 1997] Ratnaparkhi, A. (1997). A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania, Pittsburgh, PA.

[Ratnaparkhi, 1998] Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

[Ratnaparkhi et al., 1994] Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994). A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 250–255, Plainsboro, NJ.

[Raviv and Intrator, 1996] Raviv, Y. and Intrator, N. (1996). Boostrapping with noise: An effective regularization technique. *Connection Science*, 8(3-4):355–372.

[Rosen, 1996] Rosen, B. (1996). Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384.

[Rosenfeld, 1994] Rosenfeld, R. (1994). *Adaptive Statistical Language Modelling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

[Schapire, 1999] Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the Joint Conference on Artificial Intelligence*, pages 1401–1405.

[Sietsma and Dow, 1991] Sietsma, J. and Dow, R. (1991). Creating artificial neural networks that generalise. *Neural Networks*, 4:67–79.

[Tumer and Ghosh, 1996] Tumer, K. and Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–404.

[Whittemore et al., 1990] Whittemore, G., Ferrara, K., and Brunner, H. (1990). Empirical study of predictive powers of simple attachment schemes for post-modifier prepositional phrases. In *Proceedings of the 28th Annual Conference of the ACL*, pages 23–30, Pittsburgh, PA.