Language Independent Text Learning with Statistical n-Gram Language Models

by

Fuchun Peng

彭福春

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada

0-612-83015-2

Abstract

Text learning is the problem of applying machine learning to extracting useful information from large amounts of textual data. It is important in both information retrieval and statistical natural language processing. However, many current text learning systems are designed for specific languages, and/or require significant human labor to construct. In this thesis, we attempt to build language independent text learning systems that do not require significant human intervention. Our solution is based on statistical n-gram language modeling and unsupervised machine learning. Statistical language modeling is concerned with estimating the probability of word sequences, which provides a natural and principled approach to text learning. Statistical n-gram language models model text as a sequence of characters or words and offer the advantage of language independence. Unsupervised machine learning offers the advantage of significantly reducing human labor. We focus on improving performance on three text learning problems by building statistical n-gram language models and by exploiting the value of un-labeled data. These tasks include *language and task independent text classification*, *language independent lexical learning and unsupervised word segmentation*, and *Chinese text retrieval.*

The first task we consider is language and task independent text classification. Most current text classification techniques (such as naive Bayes classifiers) rely critically on feature engineering to cope with the feature explosion that occurs in text learning problems. However, feature selection is often language and task dependent, and involves many ad hoc decisions. Feature selection also tends to lose important information by discarding low frequency features. Moreover, the standard naive Bayes classifier makes independence assumptions that are too strong to model real textual data written in natural language. In particular, it assumes that observations (e.g., words) are independent of each other, given the class. We generalize the naive Bayes classifier by incorporating local dependencies between observations (characters or words) with a Markov chain. The result is a **C***hain* **A***ugmented* **N***aive Bayes* (CAN) Bayes classifier. By exploiting sophisticated smoothing techniques from language modeling research, we can deal with the feature explosion problem efficiently in CAN models. Our language modeling based approach avoids many of the ad hoc and error accumulating aspects of feature selection. The approach can work both at the character and word level. When applied at the character level, it avoids the word segmentation problem that occurs in many Asian languages. Experiments on various languages (English, Greek, Chinese, Japanese) and various text classification problems (language identification, authorship attribution, text genre detection, topic detection, sentimental classification) show it can achieve (or exceed) state of the art performance in most cases.

The second task we consider is language independent lexical learning and unsupervised word segmentation. Automated lexical learning and word segmentation is important in speech recognition and in processing the written texts of many Asian languages such as Chinese and Japanese. Traditional lexicon construction approaches are based on manually segmenting large corpora of raw data and then collecting words from the segmented corpora. Given a manually constructed lexicon, a new sequence can be segmented with heuristic methods, such as longest word match. There are some shortcomings with these traditional approaches however. For example, manually constructing a lexicon requires a huge amount of human labor. Moreover, longest word match segmentation produces fixed segmentations of a sequence, regardless of its context. To alleviate both of these limitations, we propose an unsupervised lexicon construction approach by applying an iterative optimization procedure (the EM algorithm) to n-gram models. We then segment new sequences based on dynamic programming (the Viterbi algorithm). Our approach removes the necessity of manual dictionary construction and allows for adaptive segmentations based on context. To reduce the sparse data and local maxima problems that occur with the traditional EM algorithm, we use a modified EM algorithm that exploits hierarchical structure and employs a self-supervised estimation strategy. Experiments on artificial English data and real Chinese data show the effectiveness of our methods.

Finally, we apply unsupervised word segmentation to Chinese text retrieval. Despite its low segmentation accuracy relative to supervised methods, our unsupervised segmentation method has many advantages over traditional word segmentation approaches. Experiments on the TREC-5 and TREC-6 data sets show that unsupervised word segmentation often outperforms traditional segmentation methods in Chinese text retrieval. We also find that the relationship between word segmentation performance and retrieval performance is not monotonic, as commonly expected. These findings are of theoretical and practical importance to both Chinese word segmentation and Chinese text retrieval.

Index Keywords

Language independence, Information retrieval, Statistical natural language processing, Statistical language modeling, Unsupervised machine learning, Text learning, Text classification, Lexical learning, Word segmentation, Chinese text retrieval

Acknowledgments

I am especially indebted to my supervisor Prof. Dale Schuurmans. During my four years at Waterloo, Dale has always been a very nice mentor and a considerate friend. Dale gives me full freedom to pursue my own research while still offers me very detailed guidance along the way. His broad research interests and rigorous research attitude have deeply impressed me and will continue to influence my future career. I feel very lucky to have such a great supervisor!

Deep appreciation goes to Prof. Frank Tompa. Frank always shows great enthusiasm to my research. I enjoy the discussions with Frank during our statistical NLP group meetings and seminars. His constructive comments have helped me a lot in my research. I also want to thank Prof. Charlie Clarke, Prof. Mu Zhu from the Statistics Department, and Prof. Dekang Lin from the University of Alberta for being my committee members.

I acknowledge my collaborators and co-authors: Dr. Xiangji Huang, Dr. Shaojun Wang, Prof. Vlado Keselj, Prof. Nick Cercone, Prof. Aijun An, and Prof. Stephen Robertson. I am also grateful to my fellow students: Steven Engels, Ali Ghodsi, Fletcher Lu, Paul Nijjar, Relu Patrascu, Finnegan Southey, and many other people in the statistical NLP group and AI group.

I am grateful for the financial support from Bell University Labs, MITACS, NSERC, and The University of Waterloo.

To my parents and my wife!

Bibliographic Notes

This thesis is a summary of some of the results that have already been published in international conferences and journals.

Chapter 4 is based on [110, 108, 109, 111]

Chapter 5 is based on [106] and [107].

Chapter 6 is based on the papers [62, 104, 105].

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Thesis Motivation

Nowadays, people are frustrated by being drowned in huge amounts of data while still being unable to obtain useful information. This situation is becoming worse due to the fast development of the Internet. Data exists in many formats, such as text, image, video, and audio. However, a large portion of data is in textual format; for example, books, papers, email, memos, news stories and press releases. Forrester Research (cf. [140]) has predicted that unstructured data (such as text) will become the predominant data type stored on-line. The volume of text available on the Internet is expanding very fast. As a consequence, intelligent text learning systems are in high demand. Recently, text learning has been attracting increasing interest from researchers in the areas of information retrieval, natural language processing and machine learning. The collaboration of these areas has led to significant progress in text learning. A few notable events on text learning include the KDD-2000 workshop on text mining, the IJCAI-01 workshop on text learning: beyond supervision, the ICDM-2001 workshop on text mining, the ICML-02 workshop on text learning, and the ICML-03 workshop on the continuum from labeled to unlabeled data.

Text learning is a broad concept including all learning problems related to learning from textual data. Typical examples include text retrieval, where the problem is to return documents relevant to a user query; information extraction, where the problem is to return a piece of information relevant to a query from a document; word segmentation, where the problem is to detect word boundaries in continuous speech or in written Chinese or Japanese (where no explicit word boundaries are marked); and text categorization, where the problem is to classify a document into one or more pre-defined categories. These problems are important in many real world applications. For example, text retrieval is an essential part of every search engine (such as Google, Yahoo, Infoseek, Altavista); word segmentation is the first step in speech recognition and in Chinese and Japanese information processing (such as text retrieval and machine translation); and text classification is frequently used in filtering, routing, and document organization. Improving performance on text learning problems will create direct benefit to people's lives.

One significant problem with many current text learning systems is that they (or some components of them) are designed for one specific language. Language dependence makes it difficult to adapt a system to a new linguistic environment since this requires significant re-engineering. For example, manually constructing a Chinese lexicon requires a significant amount of intervention by a person knowledgeable in Chinese. Without knowing Japanese, the same person is not able to construct a Japanese lexicon. Another aspect of language dependence is the feature selection process on which many current text classification systems critically rely. Various techniques such as stop-word removal or stemming require language specific knowledge. Moreover, many current text classification systems work on word level features. However, in many Asian languages such as Chinese or Japanese, identifying words from a character sequence itself is a difficult problem, and any word-based approach must address added complexity in coping with segmentation errors. Thus, designing language independent text learning systems would be useful in practice.

In addition to language dependence, another problem with feature selection is that it involves many ad hoc decisions and loses useful information by discarding uncommon features. There are an enormous number of possible features to consider in text categorization problems. Many standard machine learning techniques, such as naive Bayes classifiers, support vector machines, linear least squares models, neural networks, and K-nearest neighbor classifiers [152, 127], have difficulty in coping with the feature explosion. Feature engineering is critical to achieving good performance. To cope with huge amounts of features, these methods rely critically on a feature selection procedure. Once good features have been identified, almost any reasonable technique for learning a classifier seems to perform well [126]. Feature selection normally uses indirect tests, such as χ^2 or mutual information, which involve setting arbitrary thresholds and conducting a heuristic greedy search to find good feature sets and discard other features. However, the cumulative effect of uncommon features can still reduce classification accuracy, even though infrequent features contribute less information than common features individually. Consequently, throwing away uncommon features is usually not an appropriate strategy in text learning. (We will discuss this issue in detail in Chapter 2.) A better solution is to use as many features as possible. However, this causes problems for traditional classifiers since they do not cope well with an abundance of features. Efficiently coping with the feature explosion in a principled fashion is an important issue in text classification.

Lexical learning and word segmentation is another important text learning problem in natural language processing and information retrieval. Traditional lexicon construction is based on manual segmentation. A person knowledgeable in the given language first manually segments a large corpus of raw data, and then collects all words in the segmented corpus to form a lexicon. With the manually constructed lexicon, heuristic methods, such as longest word match, are then used to segment new sequences. However, there are problems with this traditional approach. One significant problem is that it requires too much human labor. Constructing a complete Chinese dictionary for word segmentation may require years of effort to segment a large corpus and collect words from the corpus. It is very useful to design an algorithm that can automatically learn a lexicon from unsegmented raw data, and use the learned lexicon for segmenting new sequences. A second problem is that the traditional longest word match produces a fixed segmentation of a sequence, regardless of its context. A better approach is segmenting sequences dynamically based on context. These two problems suggest a need for automated lexicon learning and unsupervised word segmentation. Since labeled data is not easy to obtain while unlabeled data is abundant on the Web, exploiting unlabeled data with unsupervised learning (or semi-unsupervised learning) is gaining increasing attention.

In this thesis, we build text learning systems that are language independent, remove ad hoc feature selection processes as much as possible, and do not require significant human involvement. Our goal is to design general, flexible text learning systems that are portable from language to language and domain to domain. Our solution is based on statistical language modeling, specifically, n-gram language modeling. Statistical language modeling is concerned with determining the probability of naturally occurring word (or character) sequences in a language. Thus it provides a natural and principled approach to tackling many text learning problems. N-gram models, specifically, character level n-gram models, provide an avenue for language independence.

1.2 Thesis Contributions

We use statistical n-gram language modeling as a principled solution to address the above concerns. We focus on improving three text learning problems in this thesis.

1. **Language and Task Independent Text Classification based on Statistical n-Gram Language Modeling**

   We propose the *C*hain *A*ugmented *N*aive (CAN) Bayes classifier, based on statistical n-gram language modeling, as a general text classification framework. Our approach is based on simple information theoretic principles and removes many ad hoc decisions involved in traditional classifiers. It generalizes the traditional naive Bayes classifier by allowing local Markov dependence. It can work directly at the character level or at the word level. When applied at the character level, it avoids the necessity of word segmentation in many Asian languages and provides an avenue to language independent classification. Our experimental results show that the simple approach achieves state of the art performance across a variety of languages (English, Greek, Chinese, Japanese) and tasks (language identification, authorship attribution, text genre detection, topic detection, sentimental classification) without requiring significant feature selection or extensive pre-processing.

2. **Language Independent Unsupervised Lexical Learning and Word Segmentation**

   Based on n-gram models and the EM algorithm, we propose a hierarchical EM approach for continuous speech segmentation and a self-supervised method for Chinese word segmentation. This removes the necessity of manual lexicon construction. We segment new sequences with a dynamic programming algorithm (Viterbi algorithm) based on context. To efficiently reduce the parameters, we also employ a mutual information based scheme for lexicon pruning. These techniques are shown to successfully mitigate the sparse data and local maxima problems that occur in traditional unsupervised methods.

3. **Chinese Text Retrieval with Unsupervised Word Segmentation**

   We further apply self-supervised word segmentation to Chinese text retrieval to build adaptable and robust Chinese text retrieval systems. This method combines the advantages of a traditional dictionary based approach, character based approach, and mutual information based approach, while overcoming many of their shortcomings. Experiments on TREC-5 and TREC-6 data sets show improved performance over these traditional approaches. Our method is completely language independent and unsupervised, which provides a promising avenue for constructing multi-lingual or cross-lingual information retrieval systems that are flexible and adaptive. We also investigate the relationship between word segmentation performance and retrieval performance in Chinese text retrieval. We find that, counter-intuitively, retrieval performance is not monotonically related to segmentation performance. Although the segmentation performance of the self-supervised method is not as accurate as other supervised learning systems, it nevertheless allows superior retrieval performance.

1.3 Thesis Organization

We intend to make the thesis self-contained. In addition to presenting new results, each chapter also serves as a tutorial introduction to the topic it addresses. The rest of the thesis is organized as follows. In Chapter 2 we introduce the use of n-gram models in information retrieval. In Chapter 3 we present background on statistical n-gram language modeling. In Chapter 4 we describe a language and task independent text classification approach based on statistical n-gram language modeling. In Chapter 5 we investigate an unsupervised approach to word segmentation where we present a mutual information based lexicon pruning strategy, a hierarchical EM approach for continuous speech segmentation, and a self-supervised EM algorithm for Chinese word segmentation. In Chapter 6 we investigate how self-supervised word segmentation can be used to build robust and adaptive Chinese information retrieval systems. Finally, in Chapter 7 we conclude by summarizing the thesis, pointing out ongoing research problems, and providing an outlook for future research.

Chapter 2

N-Gram Models in Information Retrieval

In this chapter, we present the concept of n-grams, and discuss past research on n-gram models in text retrieval, text classification and other applications. We demonstrate a limitation of traditional text classification approaches that employ n-gram features, which motivates us to consider a better framework for n-gram models based on statistical language modeling.

2.1 Basic Concepts

Informally, an n-gram is a concatenated sequence of n tokens (characters or words in text). Formally, given a token sequence $S = t_1 t_2 ... t_i ... t_N$, where N is the length of sequence S and t_i is an element from a finite vocabulary $\mathcal{A}$, an n-gram is defined to be any segment of S that has length n. The i^{th} n-gram of sequence S is denoted as $t_i t_{i+1} ... t_{i+n-1}$. A sequence

S of length N has $N-n+1$ n-grams; that is, S has N uni-grams, $N-1$ bi-grams, $N-2$ tri-grams, etc. [1]

Clearly, the number of possible n-grams grows exponentially with n. Let $|\mathcal{A}|$ be the size of vocabulary $\mathcal{A}$ and let $\mathcal{A}(n)$ be the number of possible n-grams. Then we have

$$\mathcal{A}(n) = |\mathcal{A}|^n$$

However, only a small portion of n-grams can be observed in any real application. This leads to the *sparse data problem.* Quoting a paragraph from [71]

> "...even for a very large data collection, the maximum likelihood estimation method does not allow us to adequately estimate probabilities of rare but nevertheless possible word sequences since many sequences occur only once; many more do not occur at all..."

To give an idea of how serious the sparse data problem is, Table 2.1 presents the statistics obtained on an English data set used in Chapter 4. The vocabulary has 95 characters, including uppercase and lowercase alphabetic characters, digits, and punctuation marks. We can see that as n grows, only a very small portion of n-grams are observed.

We will return to the sparse data problem in Chapter 3. Below we discuss research on n-gram models in information retrieval, particularly in text retrieval and text classification. Here, the concept of an n-gram model is broad. Any model that employs n-gram features could be considered as an n-gram model. The use of character n-grams offers an intuitive, yet powerful, method of representing documents. N-grams can be defined at the word

[1]For $n < 4$, Latin names are commonly used as prefixes for n-grams; for example, uni-gram, bi-gram, tri-gram. For $n \geq 4$, numeric prefixes are used instead; for example, 4-gram, 5-gram, 6-gram, etc.

n	95^n	observed number	percentage of observed n-grams (%)
1	95	95	100
2	9,025	8,684	96.2
3	857,375	332,848	38.8
4	81,450,625	1,241,920	1.5
5	7,737,809,375	2,374,071	0.03
6	735,091,890,625	3,800,519	≈ 0

Table 2.1: Sparse data problem in n-gram models

level or at the character level. However, character level n-gram models offer the following benefits and have been successfully used in many information retrieval problems.

- Language independence and simplicity: Character level n-gram models are applicable to any language, and even non-language sequences such as music or gene sequences.
- Robustness: Character level n-gram models are relatively insensitive to spelling variations and errors, particularly in comparison to word features.
- Completeness: The vocabulary of character tokens is much smaller than any word vocabulary and normally is known in advance. Therefore, the sparse data problem is much less serious in character n-gram models of the same order.

2.2 N-Gram Models in Text Retrieval

N-gram models have been extensively used in text retrieval, particularly in Asian language text retrieval, such as Chinese and Japanese. A text retrieval process is typically divided into three stages. The first stage is document indexing where content bearing terms are extracted from the document text. The second stage is index weighting which is used to enhance relevance of the documents retrieved for a query. The final stage ranks documents relative to the query according to a similarity measure. [2]

At the indexing stage, documents are normally represented as a bag of words. However, in Chinese and Japanese, words are not explicitly delimited in text. One method for avoiding word segmentation is to use n-gram character features. In Chinese, the bi-gram model is often found to be very effective. In this case, all overlapping bi-grams that occur in the documents are indexed [77], which avoids the word segmentation problem but creates a large index file and increases retrieval time. We will return to n-gram based Chinese text retrieval in Chapter 6. For English text retrieval, n-gram models are used for phrase weighting [143].

2.3 N-Gram Models in Text Classification

Many researchers have realized the importance of n-gram models for designing language independent text categorization systems [20, 35, 65]. However, they have typically used n-grams as features for a traditional feature selection process, and then deployed classifiers based on calculating feature vector similarities. Feature selection in such a classical approach is critical, and many of the required procedures, such as stop word removal, are actually language dependent. In most current text classifiers, feature engineering has been found to be critical [126].

The first attempts to employ n-gram features to obtain language independent text classifiers are presented in [20, 35]. In those work, a test document d and a class label c

[2]Three different kinds of text retrieval models have been developed so far: similarity based model (such as vector space retrieval models [122, 123]), probabilistic relevance models [117] and language modeling based models [82, 113].

are both represented by vectors of n-gram features, and a distance measure between the representations of d and c is defined. We refer this method as the ad hoc n-gram based approach. The final classification decision is made according to

$$c^* = \arg\min_{c \in C} \text{distance}(d, c) \tag{2.1}$$

Different distance metrics can be used in this approach. A simple re-ranking distance used by Cavnar and Trenkle [20] is referred to as the out-out-place (OOP) measure. In this method, a document is represented by an n-gram profile that contains selected n-grams sorted by decreasing frequencies. For each n-gram in a test document profile, we find its counterpart in the class profile and compute the number of places its location differs. The distance between a test document and a class is computed by summing the individual out-of-place values, as shown for example in Figure 2.1. For example, the 2-gram "TH" has distance 0 since it has the same rank in both the document and class profiles. The tri-gram "ING" has distance 1 since it has rank 2 in the document but has rank 3 in the class profile. The 2-gram "ED" is not found in class profile and therefore has a distance set to a default value of 1000. (We set the default value to be the maximum number of features observed.)

The standard rationale for feature selection is that current text classifiers do not cope efficiently with the feature explosion, and therefore have to rely on a feature selection procedure to reduce the number of features. However, feature selection can lose a lot of useful information by discarding uncommon features. Zipf's law [87, 155] states that, in natural language text, the frequency of a term f is inversely proportional its rank, r_f; that is,

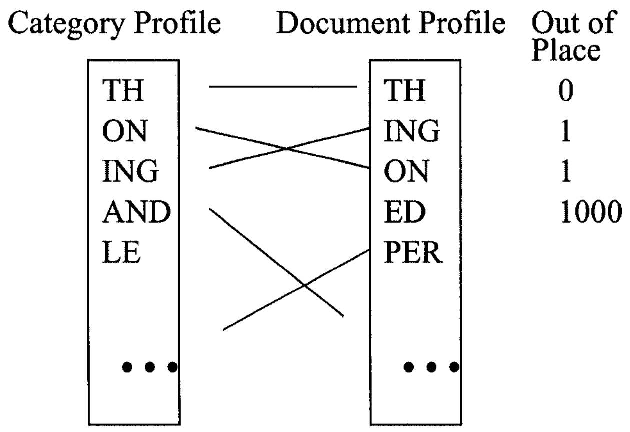

Figure 2.1: Example of an out-of-place distance calculation.

$$\log f \approx C - z \log r_f$$

where f is the frequency of a term having rank r_f, z is a positive coefficient (close to 1), and C is a constant. Figure 2.2 and Figure 2.3 show Zipf's phenomena in the English and Greek data sets used in Chapter 4. From these figures one can see that a few terms occur very frequently whereas most terms occur rarely (i.e., a heavy tail).

Zipf's law has been found to be prevalent in many domains such as natural language processing, text compression, finance and business. There are important implications of Zipf's law for text classification. One common idea is to use it for feature selection since the most frequent terms occupy a large portion of all terms [121]. However, this can be a misuse, since there is a deeper meaning of Zipf's law: most terms do not occur frequently in text, and therefore there is a large number of terms that are observed infrequently, and even a larger number of terms that are never observed at all. Thus, discarding all uncommon terms can lose a significant amount of useful information for text classification.

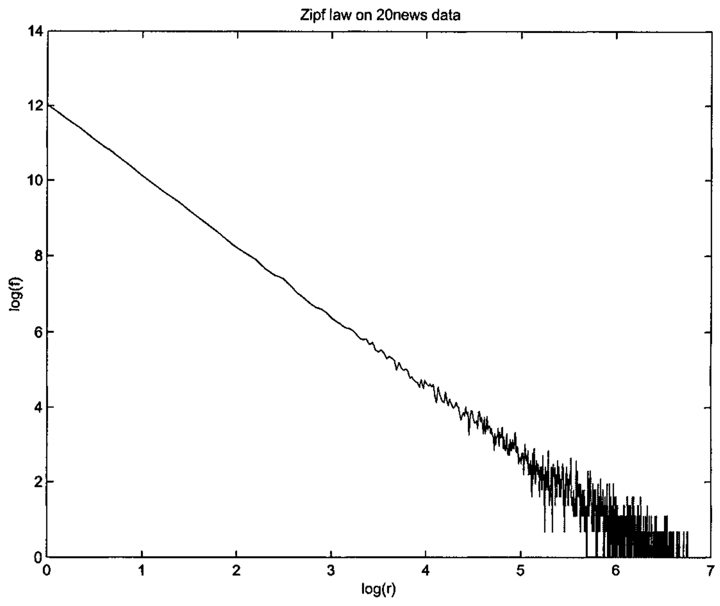

Figure 2.2: Zipf's law on English 20 news data set

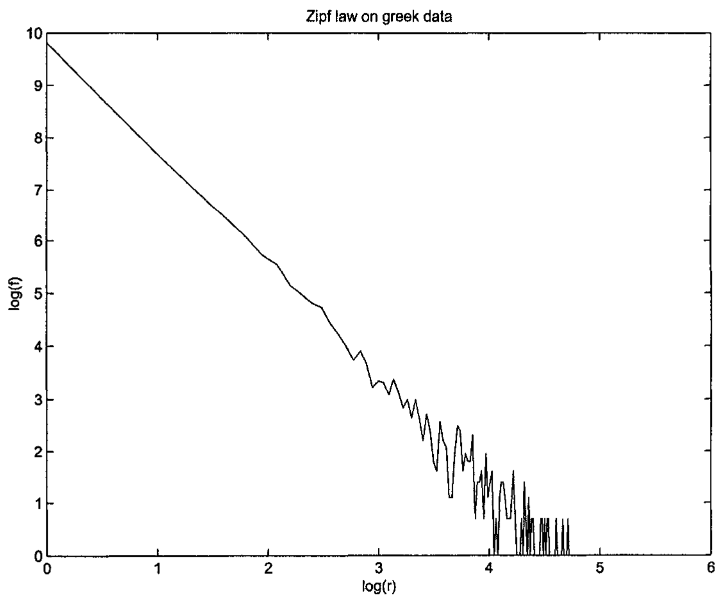

Figure 2.3: Zipf's law on Greek authorship data set

A popular feature selection technique is based on calculating information gain by determining the average mutual information of a word between all classes [33, 70]. The information gain of a word w_i is calculated as

$$\mathcal{I}(w_i, C) = \sum_{c_j} \Pr(w_i, c_j) \log \frac{\Pr(w_i, c_j)}{\Pr(w_i) P(c_j)}$$

Words are sorted by their information gain and the top ranked words are selected. Although this feature selection method works well with traditional text classifiers [90, 70], it still loses significant information by discarding uncommon features. Figure 2.4 illustrates this effect. The X-axis is the rank of features sorted according to their information gain and the Y-axis is the normalized cumulative information gain. Typically, the first 5000 features only capture about 40% of the information and therefore discarding all remaining features loses up to 60% of the information. A better way to deal with features is to use as many as possible. However, traditional text classifiers do not deal with this situation well. Another drawback of feature selection is that classifiers are sensitive to the number of selected features. Too few features cannot capture enough information, but too many features can cause over-fitting problems. Both will decrease classification performance. To obtain an optimal feature subset, normally a heuristic greedy search is used, which involves making many ad hoc decisions. Figure 2.5 illustrates the sensitivity of the ad hoc n-gram text classifier to the number of features selected, where the X-axis is the number of selected features and the Y-axis is the classification performance.

In summary, current text classifiers cannot cope efficiently with the feature explosion problem. Therefore, we should find a way to use as many features as possible, while still maintaining efficiency and effectiveness. This motivates us to consider statistical n-gram

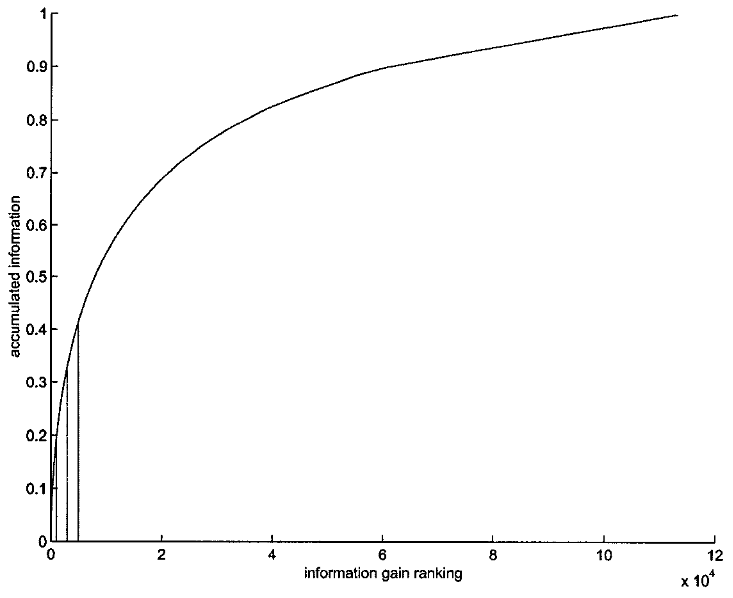

Figure 2.4: Cumulative information of features

language modeling as a general text classification approach.

2.4 N-Gram Models in Other Applications

In addition to text retrieval and text classification, n-gram models have also been successfully applied to many other domains. These include adaptive filtering [137], language identification [124], missing phoneme guessing [153], music retrieval [40] and computational immunology [88], among others.

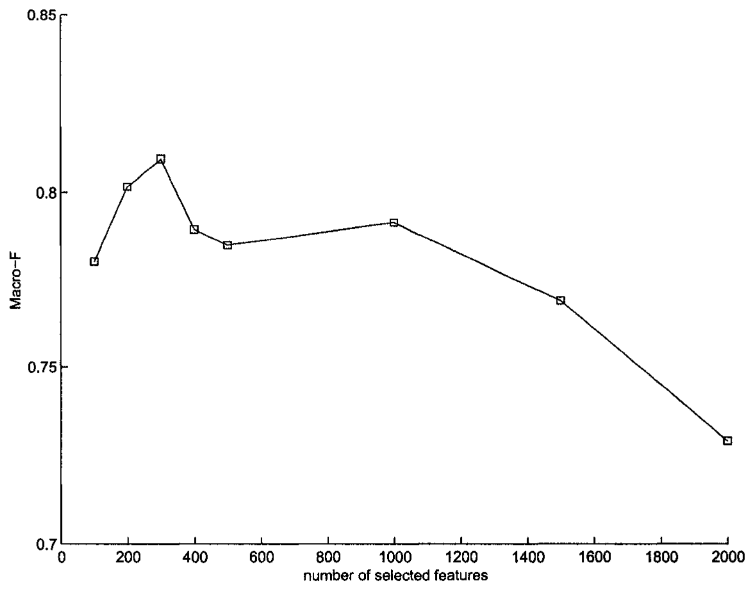

Figure 2.5: Influence of number of feature selected

2.5 Summary

In this chapter, we have introduced some of the basic concepts of n-gram models. We have also briefly reviewed past research on n-gram models for text retrieval, text classification and other applications. In particular, we examined the shortcomings of current n-gram based text classifiers, which motivates us to use a better framework to consider the feature explosion problem — statistical n-gram language modeling.

Chapter 3

Statistical n-Gram Language Modeling

In this chapter, we present some background on statistical language modeling, particularly on n-gram language models. We also briefly introduce our statistical n-gram language modeling toolkit. This chapter forms the basis for Chapter 4, where we will present the *C*hain *A*ugmented *N*aive (CAN) Bayes classifier, based on back-off statistical n-gram language modeling.

3.1 A Brief History of Language Modeling

Statistical language modeling was first investigated in the context of speech recognition. A speech recognition system is given a sequence of speech signals A, from which it attempts to recognize an uttered sequence of words W. The process of speech recognition can be formulated as follows.

$$W^* = \arg\max_W \Pr(W|A) \tag{3.1}$$

$$= \arg\max_W \Pr(A|W)\Pr(W) \tag{3.2}$$

In speech recognition, $\Pr(A|W)$ is called the acoustic model and $\Pr(W)$ is called the language model. $\Pr(W)$ computes the prior probability of a word sequence W. Without a language model, a speech recognition system will typically demonstrate poor performance. For example, the following two word sequences may have similar acoustic signals.

word sequence 1: Let's go there

word sequence 2: Let's go hair

An acoustic model alone can not discriminate between these two word sequences. However, with the introduction of language model $P(W)$, these two sentences might be easily discriminated, since "let's go there" is a much more likely word sequence than "let's go hair". Figure 3.1 illustrates the architecture of a simple speech recognition system.

In general, statistical language modeling is concerned with determining the probability of naturally occurring word sequences in a language. Although the traditional motivation for language modeling has come from speech recognition, statistical language models have recently become more widely used in many other application areas, such as information retrieval, machine translation, optical character recognition, spelling correction, document classification, information extraction, and bio-informatics.

The goal of language modeling is to predict the probability of natural word sequences, $W = w_1w_2...w_N$; or more simply, to put high probability on word sequences that actually

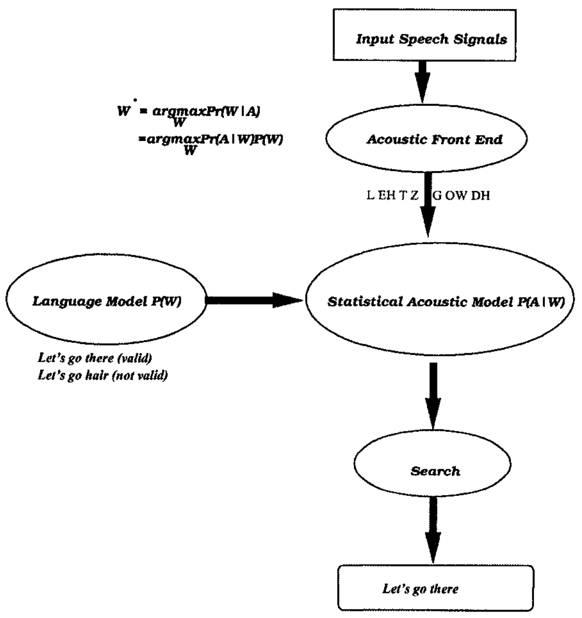

Figure 3.1: Speech Recognition

occur (and low probability on word sequences that never occur). The quality of a language model should be measured by its impact on the actual application at hand. For example, in speech recognition, the quality of a language model should be measured by how much it improves recognition accuracy. However, since speech recognition also involves a complex acoustic model, evaluating the quality of a language model in terms of recognition accuracy is complicated. In practice, people measure the quality of a language model by its empirical entropy (or perplexity) on test data. Given a word sequence $w_1 w_2 ... w_N$ to be used as a test corpus, the quality of a language model can be measured by the empirical perplexity and entropy scores on this corpus [4]

$$\begin{aligned} \textit{Perplexity} &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{\Pr(w_i|w_1...w_{i-1})}} && (3.3) \\ \textit{Entropy} &= \log_2 \textit{Perplexity} && (3.4) \end{aligned}$$

The goal is to obtain small values of these measures. Although these measures have received criticism and some new measures have been proposed [29], they are still the standard measures used in the language modeling literature.

The simplest and most successful basis for language modeling is the n-gram model wherein a word is assumed to depends only on the previous $n-1$ words. Many language models have been proposed in the literature to improve a basic n-gram models. These more sophisticated techniques include link grammars [81], sentence mixtures [67], decision trees, clustering [16], caching [68, 29], skipping models [119, 129], latent semantic analysis [8], structured language models [23, 22], neural network models [10], maximum entropy models [119, 74], and latent maximum entropy language models [146, 147]. The two references [120, 52] provide a thorough overview and systematic investigation of current techniques. However, basic language modeling research remains a hard problem. The improvements obtained by these sophisticated language models often do not justify their added complexity. In many situations, the n-gram language model is still the best choice in practice. Although basic language modeling research is hard, language modeling is attracting increasing attention recently because it has been successfully applied to many real world problems.

Below we will introduce n-gram language modeling in more detail.

3.2 N-Gram Markov Language Models

Note that by the chain rule of probability we can write the probability of any word sequence as

$$\Pr(w_1 w_2 ... w_N) = \prod_{i=1}^{N} \Pr(w_i | w_1 ... w_{i-1}) \tag{3.5}$$

An n-gram model approximates this probability by assuming that the only words relevant to predicting $\Pr(w_i | w_1 ... w_{i-1})$ are the previous $n-1$ words; that is, it assumes

$$\Pr(w_i | w_1 ... w_{i-1}) = \Pr(w_i | w_{i-n+1} ... w_{i-1})$$

A straightforward maximum likelihood estimate of n-gram probabilities from a corpus is given by the observed frequency

$$\hat{\Pr}(w_i | w_{i-n+1} ... w_{i-1}) = \frac{\#(w_{i-n+1} ... w_i)}{\#(w_{i-n+1} ... w_{i-1})} \tag{3.6}$$

where $\#(.)$ is the number of occurrences of a specified gram in the training corpus.

Although one could attempt to use these simple n-gram models to capture long range dependencies in language, attempting to do so directly immediately creates sparse data problems. Using grams of length up to n entails estimating the probability of $|\mathcal{A}|^n$ events, where $|\mathcal{A}|$ is the size of the word vocabulary. This quickly overwhelms modern computational and data resources for even modest choices of n (beyond 3 to 6). Also, because of the heavy tailed nature of language (i.e. Zipf's law) one is likely to encounter novel n-grams that were never witnessed during training in any test corpus. This is the *sparse data problem* mentioned in Chapter 2. Therefore some mechanism for assigning non-zero probability to novel n-grams is a central and unavoidable issue in statistical language modeling. The standard approach to smoothing probability estimates to cope with the sparse data problem (and to cope with potentially missing n-grams) is to use some sort of interpolated or back-off estimator [25, 71, 96, 149].

3.2.1 Smoothing Models

Linear Interpolation Model

A linear interpolated model is a linear combination of several different order n-gram models.

$$\Pr(w_i|w_{i-n+1}...w_{i-1}) = \lambda Pr_{ML}(w_i|w_{i-n+1}...w_{i-1}) + (1-\lambda)\Pr(w_i|w_{i-n+2}...w_{i-1})$$

where $\Pr_{ML}(w_i|w_{i-n+1}...w_{i-1})$ is the maximum likelihood estimate and λ is its weight which is normally computed by hold-out estimation.

Back-off Model

A back-off n-gram model [71] is defined as

$$\Pr(w_i|w_{i-n+1}...w_{i-1}) = \begin{cases} \hat{\Pr}(w_i|w_{i-n+1}...w_{i-1}), \\ \qquad \text{if } \#(w_{i-n+1}...w_i) > 0 \\ \beta(w_{i-n+1}...w_{i-1}) \times \Pr(w_i|w_{i-n+2}...w_{i-1}), \\ \qquad \text{otherwise} \end{cases} \tag{3.7}$$

where

$$\hat{\Pr}(w_i|w_{i-n+1}...w_{i-1}) = \frac{\text{discounted}\,\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})} \tag{3.8}$$

and $\beta(w_{i-n+1}...w_{i-1})$ is a normalization constant, calculated to be

$$\beta(w_{i-n+1}...w_{i-1}) = \frac{1 - \sum_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{\Pr}(x|w_{i-n+1}...w_{i-1})}{1 - \sum_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{\Pr}(x|w_{i-n+2}...w_{i-1})} \tag{3.9}$$

The derivation of the constant $\beta(w_{i-n+1}...w_{i-1})$ is shown in Appendix A.

3.2.2 Discounting Methods

Different methods can be used for computing the discounted probability (Equ. (3.8)). Typical discounting techniques include absolute smoothing, linear smoothing, Good-Turing smoothing, and Witten-Bell smoothing.

Absolute discounting

In absolute discounting, the frequency of a word is reduced by a constant c. The probability of w_i given $w_{i-n+1}...w_{i-1}$ is then calculated as:

$$\hat{\Pr}(w_i|w_{i-n+1}...w_{i-1}) = \frac{\#(w_{i-n+1}...w_i) - c}{\#(w_{i-n+1}...w_{i-1})}$$

where c is often defined as (see [97] for derivation):

$$c = \frac{n_1}{n_1 + 2n_2}$$

Here n_r denotes the number of words that occur r times. The definition of n_r also applies to the other smoothing techniques below.

⊤

Linear discounting

In linear discounting, the probability of a word w_i given $w_{i-n+1}...w_{i-1}$ is calculated as:

$$\hat{\Pr}(w_i|w_{i-n+1}...w_{i-1}) = \alpha \frac{\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})}$$

where α is defined as (see [97]):

$$\alpha = 1 - \frac{n_1}{N}$$

Here N denotes the number of events (uni-grams).

Good-Turing discounting

In Good-Turing discounting, the frequency r is discounted as (see [71, 87, 92, 93]):

$$GT_r = (r+1)\frac{n_{r+1}}{n_r}$$

where the probability of w_i given $w_{i-n+1}...w_{i-1}$ is calculated as:

$$\hat{\Pr}(w_i|w_{i-n+1}...w_{i-1}) = \frac{GT_{\#(w_{i-n+1}...w_i)}}{\#(w_{i-n+1}...w_{i-1})}$$

Witten-Bell discounting

Witten-Bell discounting [1] is similar to the linear discounting. The probability of a word w_i given $w_{i-n+1}...w_{i-1}$ is calculated as:

$$\hat{\Pr}(w_i|w_{i-n+1}...w_{i-1}) = \alpha \frac{\#(w_{i-n+1}...w_i)}{\#(w_{i-n+1}...w_{i-1})}$$

where α is defined differently as (see [149]):

$$\alpha = 1 - \frac{C}{\#(w_{i-n+1}...w_{i-1}) + C}$$

Here C denotes the number of distinct words that can follow $w_{i-n+1}...w_{i-1}$ in the training data.

3.3 Discussion

The language modeling techniques described above use individual words as the basic unit, although one could instead consider models that use individual *characters* as the basic unit. The remaining details remain the same in this case. The only difference is that the character vocabulary is always much smaller than the word vocabulary, which means that one can normally use a much higher order, n, in a character-level n-gram model (although the text spanned by a character model is still usually less than that spanned by a word model). The benefit of the character-level model is that it avoids the need for explicit word

[1]Witten-Bell smoothing is a misnomer since it was actually invented by Alistair Moffat, and is called method C in PPM text compression. We are grateful to William Teahan for pointing this out. (personal communication)

segmentation in the case of Asian languages, and thus achieves language independence. In fact, the basic unit in language modeling need not only be words or characters, it can also be other non-language units after some transformation. For example, speech can be transformed into sequences of phonemes, pictures can be transformed into sequences of 3x3 pixel combinations, musical scores can be transformed into sequences of musical symbols, Web logs can be transformed into sequences of objects [61], etc.

3.4 The Waterloo Statistical LM Toolkit

As part of the thesis, I have implemented a statistical n-gram language modeling toolkit in C++. This toolkit retains most of the functionalities of the CMU-Cambridge language modeling toolkit [30]. However, our toolkit can work at both the word level and character level to achieve language independence and domain independence in many applications and it also considers the special coding requirements of Chinese and Japanese. Use of the Waterloo LM toolkit is very similar to use of the CMU toolkit. In Appendix B we will show some of the functionalities and a typical use of the language modeling toolkit. The flowchart of a typical usage is illustrated in Figure 3.2 and an example usage is presented in Appendix B.

3.5 Summary

In this chapter, we have presented some background on Markov n-gram models, which will serve as a base for Chapter 4. As part of the thesis, I have implemented a language independent statistical n-gram language modeling toolkit.

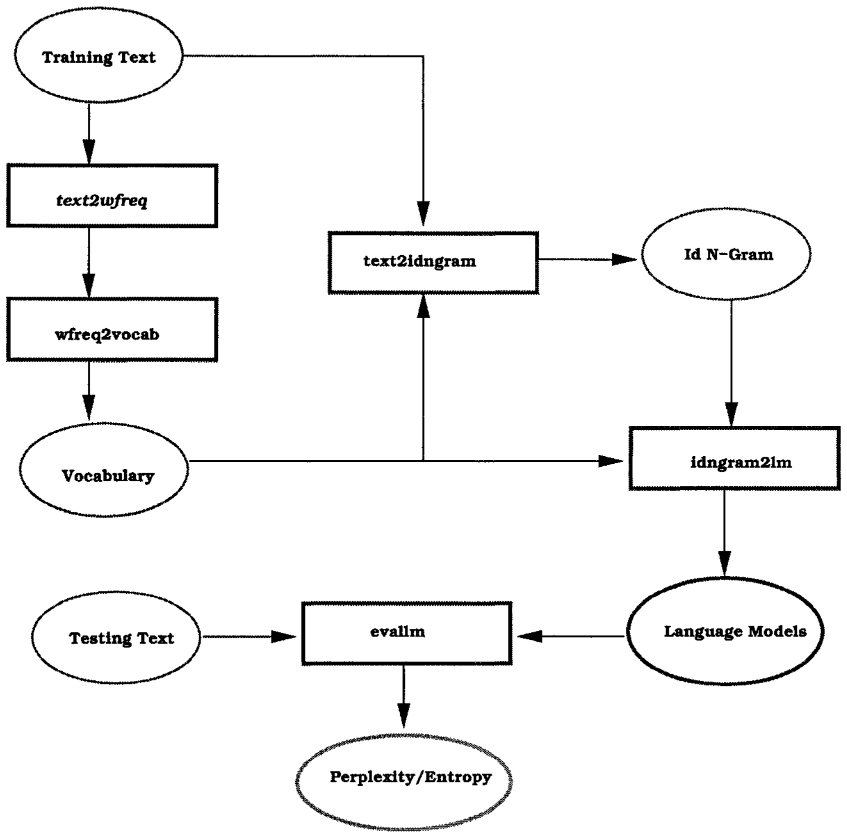

Figure 3.2: Typical usages of n-Gram language modeling toolkit

Chapter 4

Language and Task Independent Text Classification

In this chapter, we present a simple approach to language and task independent text categorization learning, based on statistical n-gram language modeling. The result is the C*hain* A*ugmented* N*aive* (CAN) Bayes classifier, a generalized naive Bayes Classifier which allows for a local Markov dependence among observations. Our approach is based on simple information theoretic principles and achieves effective performance across a variety of languages and tasks without requiring significant feature selection or extensive pre-processing. To demonstrate the language and task independence of the proposed technique, we present experimental results on several languages—Greek, English, Chinese and Japanese—in several text categorization problems—language identification, authorship attribution, text genre classification, topic detection, and sentimental classification. Our experimental results show that the simple approach achieves state of the art (or significantly better) performance in most cases.

4.1 Introduction

Text categorization concerns the problem of automatically assigning given text passages (paragraphs or documents) into predefined categories. Due to the rapid explosion of texts in digital form, text categorization has become an important area of research, owing to the need to automatically organize and index large text collections in various ways. Such techniques are currently being applied in many areas, including language identification [124], authorship attribution [135], text genre classification [73, 135], topic identification [42, 85, 90, 152], and subjective sentiment classification [142].

Many standard machine learning techniques have been applied to automated text categorization problems, such as naive Bayes classifiers, support vector machines (SVM), linear least squares fit models, neural networks, and K-nearest neighbor classifiers [152, 127]. A common aspect of these approaches is that they treat text categorization as a standard classification problem, and thereby reduce the learning process to two simple steps: feature engineering, and classification learning over the feature space. Of these two steps, feature engineering is critical to achieving good performance in text categorization problems. Once good features are identified, almost any reasonable technique for learning a classifier seems to perform well [126].

Unfortunately, the standard classification learning methodology has several drawbacks for text categorization. First, feature construction is usually language dependent. Various techniques such as stop-word removal or stemming require language specific knowledge to design adequately. Moreover, whether one can use a purely word-level approach is itself a language dependent issue. In many Asian languages such as Chinese or Japanese, identifying words from character sequences is hard, and any word-based approach must suffer added complexity in coping with segmentation errors. Second, feature selection is task dependent. For example, tasks like authorship attribution or genre classification require attention to linguistic style markers [135], whereas topic detection systems rely more heavily on bag of words features. Third, there are an enormous number of possible features to consider in text categorization problems, and standard feature selection approaches do not always cope well in such circumstances. For example, given an enormous number of features, the cumulative effect of uncommon features can still have an important effect on classification accuracy, even though infrequent features contribute less information than common features individually. Consequently, throwing away uncommon features is usually not an appropriate strategy in this domain (see Chapter 2 for detailed discussion of this). Finally, by treating text categorization as a classical classification problem, standard approaches ignore the fact that texts are written in natural language, meaning that they have many implicit regularities that can be well modeled with specific tools from natural language processing.

We will provide a simple approach based on statistical n-gram language modeling to address the above mentioned issues, resulting in the chain augmented Naive Bayes classifier. The rest of the chapter is organized as follows. We first discuss two well known text classifiers in Section 4.2, namely support vector machine (SVM) classifiers and naive Bayes classifiers. Then in Section 4.3 we present our chain augmented Naive Bayes classifier based on statistical language modeling. After that, we evaluate the chain augmented naive Bayes classifier on various languages and various text classifications tasks in Section 4.4. Finally, we present a detailed discussion and analysis in Section 4.5.

4.2 Traditional Text Classifiers

Text classification is the problem of assigning a document D to one of a set of $|C|$ pre-defined categories $C = \{c_1, c_2, ..., c_{|C|}\}$. Normally a supervised learning framework is used to train a text classifier, where a learning algorithm is provided a set of N labeled training examples $\{(d_i, c_i) : i = 1, ..., N\}$ from which it must produce a classification function $F : D \rightarrow C$ that maps documents to categories. Here d_i denotes the ith training document and c_i is the corresponding category label of d_i. We use the random variables D and C to denote the document and category values respectively. Support vector machine (SVM) classifiers and naive Bayes classifiers are currently considered the state of the art text classifiers [127].

4.2.1 Support Vector Machine Classifiers

Support vector machine (SVM) classifiers are based on Vapnik's structural risk minimization principle [144]. Given a set of N linearly separable training examples $S = \{x_i \in R^n | i = 1, 2, ..., N\}$, where each sample belongs to one of the two classes, $y_i \in \{+1, -1\}$, the SVM approach seeks the optimal hyperplane $w \cdot x + b = 0$ that separates the positive and negative examples with the largest margin. The problem can be formulated as solving the following quadratic programming problem [19, 144].

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||w||^2 \\ \text{subject to} & y_i(w \cdot x_i + b) \geq 1 \end{array} \tag{4.1}$$

Figure 4.1 gives an illustration of an SVM linear classifier.

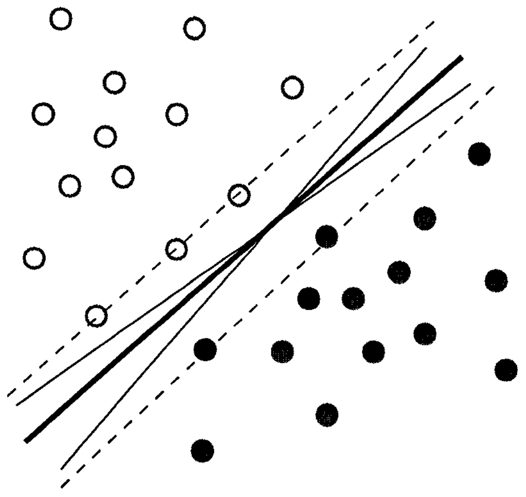

Figure 4.1: Illustration of an SVM classifier, showing examples of separating hyper-planes (solid lines), optimal hyper-plane (bold solid line) and support vectors (data on the dashed lines). The dashed line identifies the maximum margin.

The basic SVM formulation can be extended to the nonlinear case by using nonlinear kernels. Interestingly, the complexity of an SVM classifier representation does not depend on the number of features, but rather on the number of support vectors (the training examples closest to the hyperplane). This property makes SVMs suitable for large dimensional classification problems [70]. SVMs are often considered to be the state of the art classification technique in many text classification problems [152].

A shortcoming of SVM classifiers is that they do not deal with unobserved features well. They simply ignore all unobserved features by assigning them zero weight. Feature smoothing for SVMs is not as straightforward as for probabilistic models, such as naive Bayes classifiers, and remains an ongoing research problem.

4.2.2 The Naive Bayes Text Classifier

Unlike SVMs, a generative probabilistic text classifier is formulated by first postulating a joint probability model over documents d and class labels c, and then determining a classification by solving the following decision problem: given a document d, determine the class label c^* that yields the highest posterior probability $\Pr(C = c|D = d)$.

$$c^* = \arg\max_{c} \Pr(C = c|D = d) \tag{4.2}$$

Normally a generative model for text classification is formulated by a prior distribution over class labels $\Pr(c)$ and a class conditional model over documents $\Pr(d|c)$. Therefore, classification usually requires applying *Bayes' rule* [41, Chapter 10]:

$$\Pr(C = c|D = d) = \frac{\Pr(C = c) \times \Pr(D = d|C = c)}{\Pr(D = d)} \tag{4.3}$$

To simplify the presentation, we re-write Equ. (4.3) as

$$\Pr(c|d) = \frac{\Pr(c) \times \Pr(d|c)}{\Pr(d)} \tag{4.4}$$

Bayes' rule decomposes the computation of a posterior probability into the computation of a likelihood and a prior probability. In text classification, a document d is normally represented by a vector of K attributes[1] $d = (v_1, v_2, ...v_K)$. Computing $p(d|c)$ in this case is not generally trivial, since the space of possible documents $d = (v_1, v_2, ...v_K)$ is vast. To simplify this computation, the naive Bayes model introduces an additional assumption

[1]Attributes are also called features. Feature selection is an important procedure in many classifiers [126].

that all of the attribute values, v_j, are independent given the category label, c. That is, for $i \neq j$, v_i and v_j are conditionally independent given c. This assumption greatly simplifies the computation by reducing Equ. (4.4) to

$$\Pr(c|d) = \Pr(c) \times \frac{\prod_{j=1}^{K} \Pr(v_j|c)}{\Pr(d)} \tag{4.5}$$

Based on Equ. (4.5), the maximum a posterior (MAP) classifier can be constructed by seeking the optimal category which maximizes the posterior $\Pr(c|d)$:

$$\begin{aligned} c^* &= \arg\max_{c \in C} \Pr(c|d) && (4.6) \\ &= \arg\max_{c \in C} \Pr(c) \times \frac{\prod_{j=1}^{K} \Pr(v_j|c)}{\Pr(d)} && (4.7) \\ &= \arg\max_{c \in C} \Pr(c) \times \prod_{j=1}^{K} \Pr(v_j|c) && (4.8) \end{aligned}$$

Note that the step from Equ. (4.7) to Equ. (4.8) is valid because $\Pr(d)$ is irrelevant to category c. A MAP classifier (Equ. (4.2)) is optimal in the sense of minimizing *zero-one* loss (i.e. misclassification error). If the independence assumption holds, then a classifier based on Equ. (4.8) is also optimal [41].

The prior distribution $\Pr(c)$ can be computed from training data, or can also be used to incorporate additional assumptions. Two commonly used prior distributions are the Dirichlet distribution and the uniform distribution. When a uniform distribution is used as the prior, the MAP classifier becomes equivalent to the maximum likelihood (ML)

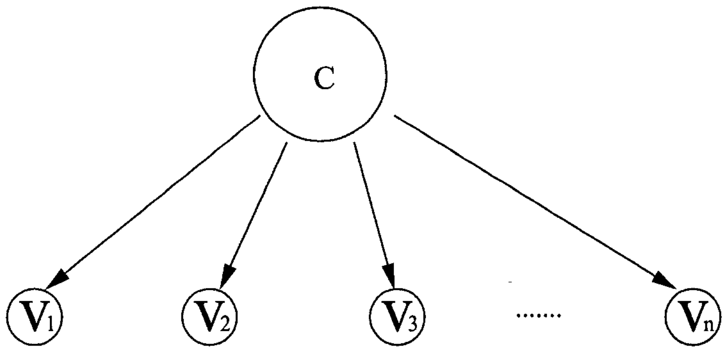

Figure 4.2: Graphical model of a naive Bayes classifier

classifier.

$$c^* = \arg\max_{c \in C} \prod_{j=1}^{K} \Pr(v_j|c) \tag{4.9}$$

Equ. (4.9) is called the *maximum likelihood* naive Bayes classifier. There are several variants of naive Bayes classifiers, including the binary independence model, the multinomial model, the Poisson model, and the negative binary independence model [44]. It has been shown that for text categorization applications, the multinomial model is most often the best choice [44, 90], therefore we will only consider the multinomial naive Bayes model in this thesis. Figure 4.2 gives a graphical representation of the multinomial naive Bayes model, showing that each attribute node is independent of the other attributes given the class label C.

The parameters of a multinomial naive Bayes classifier are given by $\Theta = \{\theta_j^c = \Pr(v_j|c) : j = 1, ..., K; c = 1, ..., |C|\}$. The likelihood of a given set of documents D^c for a given category c is given by

$$\Pr(D^c|\Theta) = \frac{N^c!}{\prod_j N_j^c!} \prod_j (\theta_j^c)^{N_j^c} \tag{4.10}$$

where N_j^c is the frequency of attribute j occurring in D^c and $N^c = \sum_j N_j^c$. A maximum likelihood estimate yields the parameter estimates

$$\theta_j^c = \frac{N_j^c}{N^c} \tag{4.11}$$

Note that Equ. (4.11) puts zero probability on the attribute values that do not actually occur in D^c (i.e., $N_j^c = 0$). Unfortunately, a zero estimate can create significant problems when we classify a new document, in particular, when we encounter a new attribute value that has not been observed in the training corpus D^c. To overcome this problem, Laplace smoothing is usually used to avoid zero probability estimates in practice:

$$\theta_j^c = \frac{N_j^c + a_j}{N^c + a} \tag{4.12}$$

where $a = \sum_j a_j$. A special case of Laplace smoothing is *add one* smoothing [87, chapter 6] obtained by setting $a_j = 1$. However, Laplace smoothing is not as effective in language modeling as some other smoothing techniques [25]. We will show that more advanced smoothing techniques can be used to improve naive Bayes classifiers, and therefore play an important role in developing effective text classifiers using naive Bayes models.

Naive Bayes classifiers have been proven successful in many domains, especially in text classification [86, 90, 117], despite the simplicity of the model and the restrictiveness of the independence assumptions it makes. Domingos and Pazzanni [39] point out that naive Bayes classifiers can obtain near optimal misclassification error even when the independence assumption is strongly violated. Nevertheless, it is commonly thought that relaxing the independence assumption of naive Bayes ought to allow for superior text classification [86], and it has been shown in practice that functionally dependent attributes can indeed improve classification accuracy in some cases [47, 116].

A significant amount of research has been conducted on relaxing the naive Bayes independence assumption in machine learning research. A well known extension is the **T***ree* **A***ugmented* **N***aive* Bayes classifier (TAN) [47] which allows for a tree-structured dependence among observed variables in addition to the traditional dependence on the hidden "root" variable. However, learning tree structured Bayesian networks is not trivial [72], and this model has rarely been used in text classification applications. We will investigate a convenient alternative that lies between pure naive Bayes and TAN Bayes models in the strength of its assumptions; namely, the **C***hain* **A***ugmented* **N***aive Bayes* (CAN) model. A CAN Bayes model simplifies the TAN model by restricting dependencies among observed variables to form a Markov chain instead of a tree. Interestingly, it turns out that the model that results is closely related to n-gram language models which have been widely studied in statistical natural language modeling and speech recognition. Below, we will augment the naive Bayes text classifier by including attribute dependencies that form a Markov chain, and use techniques from statistical n-gram language modeling to learn and apply these models. The result is a combination of naive Bayes and statistical n-gram methods that yields simple yet surprisingly effective classifiers.

4.3 Language Models as Text Classifiers

Our approach to applying language models to text categorization is to use Bayesian decision theory, similar to the naive Bayes model. Assume we wish to classify a text $d = w_1 w_2 .... w_N$ into a category $c \in C = \{c_1, ..., c_{|C|}\}$. A natural choice is to pick the category c that has the largest posterior probability given the text. That is,

$$c^* = \arg\max_{c \in C} \Pr(c|d) \tag{4.13}$$

Using Bayes rule, this can be rewritten as

$$\begin{aligned} c^* &= \arg\max_{c \in C} \Pr(c) \Pr(d|c) && (4.14) \\ &= \arg\max_{c \in C} \Pr(c) \prod_{i=1}^{N} Pr_c(w_i|w_{i-n+1}...w_{i-1}) && (4.15) \end{aligned}$$

Here, $\Pr(d|c)$ is the likelihood of d under category c, which can be computed by n-gram language modeling. The likelihood is related to perplexity and entropy by Equ. (3.3) and Equ. (3.4). The prior $\Pr(c)$ can be computed from training data or can be used to incorporate more assumptions, such as a uniform or Dirichelet distribution. $\Pr_c(w_i|w_{i-n+1}...w_{i-1})$ is computed using a back-off model:

$$Pr_c(w_i|w_{i-n+1}...w_{i-1}) = \begin{cases} \hat{\Pr}_c(w_i|w_{i-n+1}...w_{i-1}), \\ \qquad \text{if } \#_c(w_{i-n+1}...w_i) > 0 \\ \beta_c(w_{i-n+1}...w_{i-1}) \times \Pr_c(w_i|w_{i-n+2}...w_{i-1}), \\ \qquad \text{otherwise} \end{cases} \tag{4.16}$$

where $\#_c(w_{i-n+1}...w_i)$ is the number of times of $w_{i-n+1}...w_i$ occurs in all training documents of category c, $\hat{\Pr}_c(w_i|w_{i-n+1}...w_{i-1})$ is the discounted probability calculated as

$$\hat{Pr}_c(w_i|w_{i-n+1}...w_{i-1}) = \frac{\text{discounted}\ \#_c(w_{i-n+1}...w_i)}{\#_c(w_{i-n+1}...w_{i-1})} \tag{4.17}$$

and $\beta_c(w_{i-n+1}...w_{i-1})$ is a normalization constant, calculated to be

$$\beta_c(w_{i-n+1}...w_{i-1}) = \frac{1 - \sum\limits_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{Pr}_c(x|w_{i-n+1}...w_{i-1})}{1 - \sum\limits_{x:\#(w_{i-n+1}...w_{i-1}x)>0} \hat{Pr}_c(x|w_{i-n+2}...w_{i-1})} \tag{4.18}$$

Therefore, our approach is to learn a separate back-off language model for each category, by training on a data set from that category. Then, to categorize a new text d, we supply d to each language model, evaluate the likelihood (or entropy) of d under the model, and pick the winning category according to Equ. (4.15). Figure 4.3 illustrates this procedure. The parameters in the model are $\Theta = \{\theta_i^c = \hat{\Pr}_c(w_i|w_{i-n+1}...w_{i-1}), \beta_i^c = \beta_c(w_{i-n+1}...w_{i-1}) : i = 1, ..., N; c = 1, ..., |C|\}$.

The inference of an n-gram based text classifier is very similar to a naive Bayes classifier. In fact, n-gram classifiers are a straightforward generalization of naive Bayes: a uni-gram classifier with Laplace smoothing corresponds exactly to the traditional naive Bayes classifier. However, n-gram language models, for larger n, possess many advantages over naive Bayes classifiers, including modeling longer context and applying superior smoothing techniques in the presence of sparse data. Another notable advantage of the n-gram language modeling based approach is that it does not have an explicit feature selection procedure. Instead, it considers all possible n-grams as features and their importance

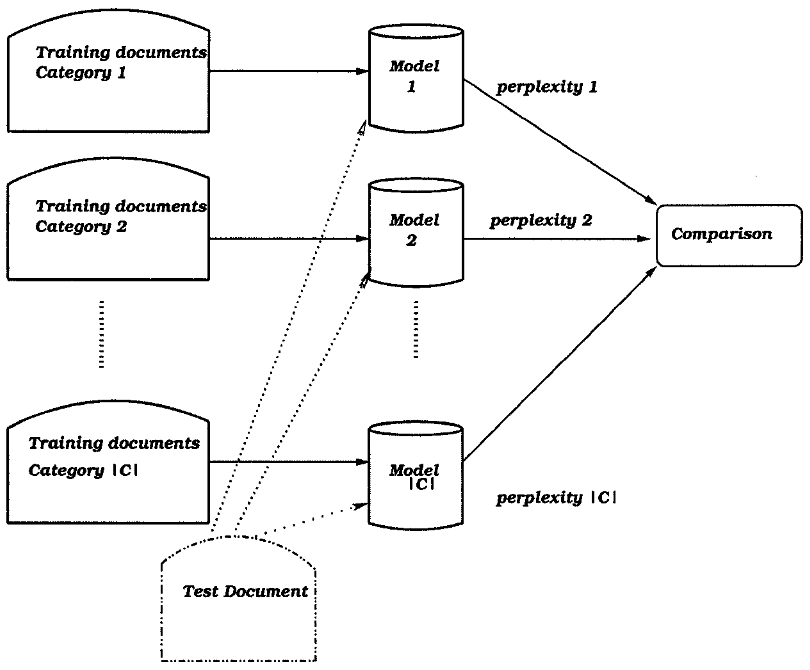
Training documents Category 1
Training documents Category 2
Training documents Category |C|
Model 1
Model 2
Model |C|
perplexity 1
perplexity 2
perplexity |C|
Comparison
Test Document

Figure 4.3: Illustration of LM based text classification

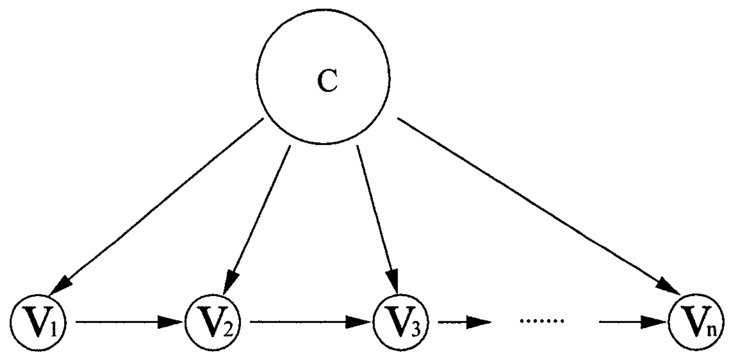

Figure 4.4: Graphical model of a bi-gram chain augmented naive Bayes classifier

is implicitly considered by their contribution to the quality of language modeling (in the sense of perplexity given in Equ. (3.3)). The back-off smoothing technique effectively deals with the feature explosion.

In a naive Bayes text classifier, attributes (words) are considered to be independent of each other given the category. However, in a language modeling based approach, this is enhanced by considering a Markov dependence between adjacent words. Due to this similarity, we refer the n-gram augmented naive Bayes classifier as a **C***hain* **A***ugmented* **N***aive Bayes* classifier (CAN). A graphical model of a bi-gram augmented naive Bayes text classifier is given in Figure 4.4, where each leaf node is a word occurring sequentially in a document.

4.4 Experimental Comparison

We now proceed to present our results on several text categorization problems in different languages. Specifically, we consider language identification, authorship attribution (on Greek, Chinese and English), Greek genre classification, topic detection (English, Chinese, and Japanese) and sentimental classification.

4.4.1 Measuring Classification Performance

In many cases, classification can be carried out directly based on Equ. (4.2).[2] However, some classifiers such as SVMs, are specifically designed only for binary classification problems. For these classifiers, even given a single $|C|$ category classification problem, we have to convert it into a set of $|C|$ binary classification problems. Probabilistic classifiers can be directly used for classifying into $|C|$ categories based on Equ. (4.2).

Different performance measures can be used to assess these two different unique classification approaches. For the sake of consistency with previous research, we experiment with both approaches in different data. In the Chinese topic detection experiments, there are 6 classes, so we formulate 6 binary classification problems. In Reuters topic detection experiments, there are 10 classes, so we formulate 10 binary classification problems. In both of these cases, we measure classification performance by *micro-averaged F-measure.* To calculate the micro-averaged score, we formed an aggregate confusion matrix by adding up the individual confusion matrices from each category. The micro-averaged precision, recall, and F-measure can then be computed based on the aggregated confusion matrix.

In other experiments, we measured *overall accuracy* and *macro-averaged F-measure.* (Micro-averaged values are not available in these cases.) Here the precision, recall, and F-measures of each individual category can be computed based on a $|C| \times |C|$ confusion matrix. Macro-averaged scores can be computed by averaging the individual scores. The

[2]In most problems we consider, each text can only belong to a unique category. However, in some problems, such as Reuters-21578 data, an individual text can simultaneously belong to several categories (multi-way classification). A multi-way classification problem is normally converted into multiple unique classification problems, one for each possible category.

overall accuracy is computed by dividing the number of correctly identified documents (summing the numbers across the diagonal) by the total number of test documents.

4.4.2 Language Identification

The first text categorization problem we examined was language identification—a useful pre-processing step in multi-lingual information retrieval. In our experiments, we considered one chapter of the Bible that had been translated into 6 different languages: English, French, German, Italian, Latin and Spanish. In each case, we reserved twenty sentences from each language for testing and used the remainder for training. For this task, with bi-gram or greater character-level models and all of the available smoothing techniques, we achieved **100%** accuracy.

Language identification is probably the easiest text classification problem because of the significant morphological differences between languages,[3] even when they are based on the same character set. Other techniques can also achieve accurate performance [124].

4.4.3 Authorship Attribution

The second text categorization problem we examined was author attribution. A famous example is the case of the *Federalist Papers*, of which twelve instances are claimed to have been written both by Alexander Hamilton and James Madison [59]. Authorship attribution is more challenging than language identification because the difference among authors is much more subtle than that among different languages. To demonstrate the language independence of our approach, we experiment on three different languages: Greek, English

[3]Language identification from speech is of course much harder.

Code	Author Name	Train size (characters)
B0	S. Alaxiotis	77295
B1	G. Babiniotis	75965
B2	G. Dertilis	66810
B3	C. Kiosse	102204
B4	A. Liakos	89519
B5	D. Maronitis	36665
B6	M. Ploritis	72469
B7	T. Tasios	80267
B8	K. Tsoukalas	104065
B9	G. Vokos	64479

Table 4.1: Authors in the Greek authorship attribution data set

and Chinese.

Greek data sets

We considered a data set used by Stamatatos et al. [135] consisting of 20 texts written by 10 different modern Greek authors (totaling 200 documents). In each case, 10 texts from each author were used for training and the remaining 10 for testing. The specific authors that appear are shown in Table 4.1.

The results are shown in Table 4.2. In our experiments, we obtained the best performance by using a tri-gram model with absolute smoothing. The best accuracy we obtained is **90%**. This compares favorably to the best accuracy reported by Stamatatos et al. [135] of 72%. [4] The 18% accuracy improvement is surprising given the relative simplicity of our method.

To compare the individual author categories, Table 4.3 gives the confusion matrix.

[4]Note that Stamatatos et al.'s measures of *identification error* and *average error* correspond to our *recall* and *overall accuracy* measures respectively.

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.57	0.53	0.56	0.51	0.55	0.49	0.55	0.49	0.55	0.49
2	0.85	0.84	0.82	0.79	0.80	0.75	0.84	0.83	0.84	0.82
3	**0.90**	0.89	0.70	0.64	0.79	0.72	0.89	0.88	0.89	0.87
4	0.87	0.85	0.47	0.39	0.79	0.72	0.85	0.82	0.88	0.86
5	0.86	0.85	0.35	0.29	0.79	0.72	0.87	0.85	0.86	0.83
6	0.86	0.83	0.16	0.10	0.79	0.73	0.87	0.85	0.86	0.83

Table 4.2: Results on Greek authorship attribution using character models

Comparing Table 4.3 to the original results [135, Table 6], shows that we obtain better results in every category.

As a typical western language, Greek words are separated by white-spaces. We can therefore also apply word level models to the Greek data. Table 4.4 shows the word level results. The performmance goes up to 96%, which is better than character level models. However, such improvement is not always observed as we will see in other experiments.

English data set

The English data used in our experiments is available from the Alex Catalogue of Electronic Texts.[5] We used the 8 most prolific authors from this collection, shown in Table 4.5.

To reduce any sparse data problems we might face, we first converted the corpus to lowercase characters and only used the 30 most frequent characters in the vocabulary, which comprises over 99% of the character occurrences in the corpus. The best accuracy we obtained was **98%**, which was achieved by a 6-gram model using absolute smoothing. This is excellent performance. However, it is probably due to the distinct writing styles of

[5]http://www.infomotions.com/alex/downloads/

	Computer Estimate										
True Label	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	Recall
B0	8								2		0.8
B1		10									1.0
B2			8				1			1	0.8
B3				10							1.0
B4					10						1.0
B5		2			3	4			1		0.4
B6							10				1.0
B7								10			1.0
B8									10		1.0
B9										10	1.0
Precision	1.00	0.83	1.00	1.00	0.77	1.00	0.91	1.00	0.77	0.91	
Overall Accuracy: **0.90**						Macro-average F-measure: **0.89**					

Table 4.3: Confusion matrix on the Greek authorship data using tri-gram model with absolute smoothing

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	**0.96**	0.95	0.76	0.72	0.93	0.92	**0.96**	0.95	0.61	0.57
2	**0.96**	0.95	0.60	0.56	0.95	0.94	0.95	0.94	0.83	0.78
3	**0.96**	0.95	0.54	0.51	0.95	0.94	0.95	0.94	0.83	0.78
4	**0.96**	0.95	0.56	0.53	0.95	0.94	0.95	0.94	0.84	0.80

Table 4.4: Results on Greek authorship attribution using word level models

	Author Name	Training size word(character)
E0	Charles Dickens	1614258 (9033267)
E1	John Keats	49314 (335676)
E2	John Milton	146446 (868857)
E3	William Shakespeare	645605 (3642829)
E4	Robert L. Stevenson	1036108 (5687003)
E5	Oscar Wilde	102092 (585092)
E6	Ralph W. Emerson	384570 (2201546)
E7	Edgar Allan Poe	307710 (1785067)

Table 4.5: Authors appearing in the English authorship attribution data set.

these famous authors.

Chinese data

Authorship attribution in Chinese normally requires an initial word segmentation phase, followed by a feature extraction process at the word level, as in English. However, word segmentation is itself a hard problem in Chinese, and an improper segmentation may cause insurmountable problems for later prediction phases. We avoid the word segmentation problem by simply operating at the character level.

The Chinese corpus we used in our experiments was also downloaded from the Internet.[6] Eight of the most popular modern Chinese martial art novelists were included in this study, shown in Table 4.6. One or two novels was selected from each author to be used as training data, and an additional 20 novels were used as a test set.

Note that a significant difference between Chinese and English (or Greek) is that the Chinese character vocabulary is much larger than the English (or Greek) character vo-

[6]http://chineseculture.about.com/library/chinese/blindex.htm

	Author Name	Training size (character)
C0	Gu Long	1466286
C1	Huang Yi	1860555
C2	Jin Yong	979885
C3	Liang Yusheng	1085179
C4	Wen RuiAn	536986
C5	Xiao Yi	875678
C6	Chen Qinyun	857929
C7	Wo Losheng	1554689

Table 4.6: Authors appearing in the Chinese authorship attribution data.

cabularies. For example, the most commonly used Chinese character set contains 6763 characters. In our experiments, we encountered about 4600 distinct Chinese characters. Therefore, to reduce the sparse data problems, we first selected the most frequent 2500 characters as our vocabulary, which comprised about 99% of all character occurrences.

The best overall accuracy we obtained was **94%**, with a tri-gram language model using Witten-Bell smoothing. Once again, this is effective performance, but possibly obtained on an easy data set.

4.4.4 Text Genre Classification

The third problem we examined was text genre classification, which is an important application in natural language processing and information retrieval [73, 84]. We considered again a Greek data set used by Stamatatos et al. [135] consisting of 20 texts of 10 different styles extracted from various sources (200 documents in total). For each style, we used 10 texts as training data and the remaining 10 as testing data. The 10 different text genres and their original sources are shown in Table 4.7.

code	Genre	Source
0	Press editorial	Newspaper TO BHMA
1	Press reportage	Newspaper TO BHMA
2	Academic prose	Journal of archives of hellonic pathology
3	Official documents	High court decisions, ministerial decisions
4	Literature	Various pages
5	Recipes	Magazine NETLIFE
6	Curriculum Vitae	Various pages
7	Interviews	Newspaper TO BHMA
8	Planned speeches	Ministry of defense
9	Broadcast news, scripted	Radio station, FLASH 9.61

Table 4.7: 10 Different Text Genres

The results of learning an n-gram based text classifier are shown in Table 4.8. The **86%** accuracy obtained with bi-gram models compares favorably to the 82% reported earlier [135], which again is based on a much deeper NLP analysis.

Again, we can apply word level models to Greek text genre classification. We obtain 81% accuracy using absolute smoothing and $n = 1$. Here the word level models are worse than the character level models. Below, we will see more cases where character level models are superior to word level models.

4.4.5 Topic Detection

The fourth problem we examined was topic detection in text, which is a heavily researched text categorization problem [42, 85, 90, 152, 127]. Here we demonstrate the language independence of the character based language modeling approach by considering experiments on English, Chinese and Japanese data sets.

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.31	0.55	0.30	0.54	0.30	0.54	0.30	0.54	0.30	0.54
2	**0.86**	0.86	0.72	0.69	0.60	0.52	0.82	0.81	**0.86**	0.86
3	0.77	0.75	0.50	0.41	0.65	0.59	0.79	0.77	0.85	0.85
4	0.69	0.65	0.38	0.29	0.58	0.50	0.74	0.69	0.76	0.74
5	0.66	0.61	0.38	0.29	0.56	0.49	0.69	0.66	0.73	0.70
6	0.62	0.57	0.38	0.29	0.49	0.53	0.67	0.63	0.71	0.68
7	0.63	0.58	0.33	0.21	0.49	0.53	0.66	0.62	0.70	0.68

Table 4.8: Results on Greek text genre classification using character models

English Data

The English 20 Newsgroup data was originally collected and used for text categorization by Lang [83] and has been widely used in topic detection research [90, 115].[7] This collection consists of 19,974 non-empty documents distributed evenly across 20 newsgroups. We use the newsgroups to form our categories, and randomly select 80% of the documents to be used for training and set aside the remaining 20% for testing. Table 4.9 shows the 20 categories appearing in the newsgroup data.

We considered text to be a sequence of characters, and learned character-level n-gram models. The resulting classification accuracies are reported in Table 4.10. We can also consider English text as a sequence of words and therefore also apply word models in this case. The results are shown in Table 4.11. All results are averaged across 5 random runs. In the character model, with tri-gram (or higher order) models, we consistently obtain accurate performance, peaking at **89.13±0.33%** accuracy in the case of 6-gram models with Witten-Bell smoothing. We note that word-level models were able to achieve

[7]http://www.ai.mit.edu/˜ jrennie/20Newsgroups/

Code	Newsgroup	Code	Newsgroup
0	alt.atheism	10	rec.sport.hockey
1	comp.graphics	11	sci.crypt
2	comp.os.ms-windows.misc	12	sci.electronics
3	comp.sys.ibm.pc.hardware	13	sci.med
4	comp.sys.mac.hardware	14	sci.space
5	comp.windows.x	15	soc.religion.christian
6	misc.forsale	16	talk.politics.guns
7	rec.autos	17	talk.politics.mideast
8	rec.motorcycles	18	talk.politics.misc
9	rec.sport.baseball	19	talk.religion.misc

Table 4.9: English 20 Newsgroup data

88.22±0.35% accuracy in this case. (Using the t-test, we find that the difference between the results of character level models and word level models are statistically significant to the $\alpha = 0.01$ significance level. The associated p-value is 0.0027 and t-score is 4.2818.) These results compare favorably to the state of the art result of 87.5% accuracy reported elsewhere [115], which was based on a combination of SVMs with error correcting output coding (ECOC).

Note that the word level model with Laplace smoothing and $n = 1$ is exactly equivalent to the traditional Naive Bayes classifier. The accuracy is 84.93%, which is consistent with previous findings [90].

Japanese Data

Similar to Chinese, Japanese topic detection is often thought to be more challenging than in English, because words are not white-space delimited in Japanese text. This fact seems to require word segmentation to be performed as a pre-processing step before further

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.2128	0.2034	0.2128	0.2033	0.2126	0.2032	0.2126	0.2032	0.2126	0.2032
2	0.6691	0.6505	0.6646	0.6451	0.6760	0.6569	0.6772	0.6584	0.6628	0.6437
3	0.8680	0.8635	0.8458	0.8385	0.8617	0.8571	0.8629	0.8596	0.8673	0.8621
4	0.8879	0.8849	0.8279	0.8197	0.8803	0.8775	0.8783	0.8760	0.8903	0.8870
5	0.8895	0.8867	0.7384	0.7267	0.8803	0.8764	0.8824	0.8806	0.8923	0.8896
6	0.8876	0.8852	0.6517	0.6453	0.8787	0.8759	0.8810	0.8794	**0.8913**	0.8883
7	0.8862	0.8831	0.5800	0.5759	0.8771	0.8742	0.8838	0.8821	0.8909	0.8875

Table 4.10: Topic detection results on English 20 Newsgroup data using character models

n	Absolute		Laplace		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.8735	0.8694	*0.8493*	0.8412	0.8653	0.8612	0.8757	0.8717	0.8423	0.8343
2	0.8818	0.8781	0.4730	0.4718	0.8733	0.8686	**0.8822**	0.8790	0.8749	0.8711
3	0.8800	0.8766	0.3381	0.3416	0.8713	0.8668	0.8798	0.8769	0.8726	0.8693
4	0.8808	0.8774	0.3234	0.3274	0.8712	0.8667	0.8795	0.8766	0.8730	0.8699

Table 4.11: Topic detection results on English 20 Newsgroup data using word models

n	Absolute		Good-Turing		Linear		Witten-Bell	
	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac	Acc.	F-Mac
1	0.33	0.29	0.34	0.29	0.34	0.29	0.34	0.29
2	0.66	0.62	0.66	0.61	0.66	0.63	0.66	0.62
3	0.75	0.72	0.75	0.72	0.76	0.73	0.75	0.72
4	0.81	0.77	0.81	0.76	0.82	0.76	0.81	0.77
5	0.83	0.77	0.83	0.76	0.83	0.76	0.83	0.77
6	**0.84**	0.76	0.83	0.75	0.83	0.75	**0.84**	0.77
7	**0.84**	0.75	0.83	0.74	0.83	0.74	**0.84**	0.76
8	0.83	0.74	0.83	0.73	0.83	0.73	**0.84**	0.76

Table 4.12: Japanese topic detection results

classification [1]. However, we avoid the need for explicit segmentation by simply using a byte-level n-gram classifier.

We consider the Japanese topic detection data investigated by Aizawa [1]. This data set was converted from the NTCIR-J1 data set originally created for Japanese text retrieval research. The data has 24 categories. The testing set contains 10,000 documents distributed unevenly between categories. We have obtained the experimental results shown in Table 4.12, which still show an **84%** accuracy rate on this problem (for 6-gram or higher order models). This is the same level of performance as that reported in [1], which uses an SVM approach with word segmentation, morphology analysis and feature selection.

Chinese Data

Chinese poses the same word segmentation issues as Japanese. Word segmentation is also thought to be necessary for Chinese text categorization [55], but we avoid the need again by considering character level language models.

For Chinese topic detection we considered a data set investigated in He et al. [55]. The

Code	Topic
0	Politics, Law and Society
1	Literature and Arts
2	Education, Science and Culture
3	Sports
4	Theory and Academy
5	Economics

Table 4.13: Chinese topics

corpus in this case is a subset of the TREC-5 data set created for research on Chinese text retrieval. To make the data set suitable for text categorization, documents were first clustered into 101 groups that shared the same headline (as indicated by an SGML tag) and the six most frequent groups were selected to make a Chinese text categorization data set. Following earlier researchers [55, 56], we converted the problem into 6 binary classification problems. In each case, we randomly selected 500 positive examples and then selected 500 negative examples evenly from among the remaining negative categories to form the training data. The testing set contains 100 positive documents and 100 negative documents generated in the same way. The training set and testing set do not overlap and do not contain repeated documents.

Table 4.14 shows the results of the character level language modeling classifiers. Table 4.15 shows the results of using an SVM classifier. (We use the SVM^{light} [70] toolkit with linear kernels.) The entries are micro-average F-measure. We observe a performance of over **86.7%** for this task, using bi-gram (2 Chinese characters) or higher order models. SVM achieves stable performance of about 81.5%. We can see that SVM is robust to the number of features in this case.

Note that Japanese and Chinese characters are both encoded with 2 bytes in a computer.

	Absolute	Laplace	Good-Turing	Linear	Witten-Bell
1	0.855	*0.857*	0.858	0.859	0.859
2	**0.867**	0.798	0.862	0.860	0.859
3	0.864	0.794	0.859	0.859	0.862
4	0.866	0.801	0.860	0.863	0.862

Table 4.14: Results of character level language modeling classifier on Chinese topic detection data.

Feature #	Micro-F1
100	0.811
200	0.813
300	0.817
400	0.816
500	0.817
1000	0.817
1500	0.815
2000	0.816

Table 4.15: Results of the character level SVM classifier on Chinese data.

A sequence of Chinese characters can be considered as a concatenated sequence of either one-byte characters or two-bytes characters. In the Chinese experiments, we considered this coding requirement (a slightly worse performance was obtained without this consideration due to noise). However, in the Japanese experiments, we did not consider this requirement and merely used byte-level n-gram models. The reason is that we have no knowledge of Japanese and could not guarantee the correctness of character segmentation in this case. This explains why the optimal context length in the Japanese results (Table 4.12) is about twice that of the Chinese results (Table 4.14).

Reuters-21578 data experiments

Unlike other data sets, Reuters-21578 data is used for multiway classification, where each document is allowed to belong to more than one category.[8] The data set contains 12902 Reuters news wire articles. The documents are assigned to 135 topic categories. However, some categories are empty and thus there are only 118 non-empty categories, among which the 10 most frequent categories contain about 75% of the documents. There are several ways to split the documents into training and testing sets: 'ModLewis' split, 'ModApte' split, and 'ModHayes' split. The 'ModApte' train/test split is widely used in text classification research. Using the 'ModApte' split, the 10 most frequent categories and the numbers of documents used for training, testing and reserved unused in each category are listed in Table 4.16.

The results of micro-averaged F-measure are shown in Table 4.17 and Table 4.18 for character level models and word level models respectively.

[8]The data set is publicly available at http://www.research.att.com/~ lewis

category	training	test	unused
earn	2877	1087	22
acq	1650	719	79
money-fx	538	179	82
grain	433	149	45
crude	389	189	54
trade	369	118	65
interest	347	131	33
wheat	212	71	23
ship	197	89	18
corn	182	56	15

Table 4.16: Reuters top 10 most frequent categories

n	Absolute	Laplace	Good-Turing	Linear	Witten-Bell
1	0.4213	0.4207	0.4207	0.4207	0.4207
2	0.6607	0.6731	0.6790	0.6651	0.6651
3	0.7816	0.8032	0.7873	0.7883	0.7805
4	0.8010	0.7229	0.8024	0.7969	0.8023
5	0.7964	0.6448	0.7932	0.7855	0.7996
6	0.7832	0.6156	0.7839	0.7765	0.7884
7	0.7701	0.6131	0.7782	0.7697	0.7813

Table 4.17: Results of the character level language modeling classifier on Reuters data

n	Absolute	Laplace	Good-Turing	Linear	Witten-Bell
1	0.7850	0.7973	0.7900	0.7880	0.7959
2	0.8152	0.7069	0.8124	0.8114	0.7935
3	0.8137	0.6353	0.8135	0.8105	0.7952

Table 4.18: Results of the word level language modeling classifier on Reuters data

Word level models marginally outperform character level models. Bi-gram or tri-gram word level models are better than uni-gram models, which means that relaxing the independence assumption in naive Bayes models also helps in this case. However, in previous research, an SVM classifier obtained up to 92% accuracy [42], which was based on feature selection and optimizing prior $\Pr(c)$ for each individual category. We did not optimize the prior information and merely used uniform prior (that is, we put equal weight for positive and negative category).

4.4.6 Sentimental Classification

The last text classification problem we examined is sentimental classification, an emerging text classification task which judges the sentimental view of a given text. For example, given a article from CNN, is it positive or negative for Bush's policy? Are the comments subjective or objective?

We use a set of movie review data originally constructed by Pang and Lee [103].[9] The data set consists of 700 positive and 700 negative texts. Following [103], we use 3 fold cross validation testing. The best results reported in [103] are 82.9% accuracy for an SVM and 78.7% for naive Bayes. For this problem, we obtain 81.5% using word level language models and 73% using character level language models.

[9]Available on-line: http://www.cs.cornell.edu/people/pabo/movie-review-data/

4.5 Analysis and Discussion

The perplexity of a test document under a language model depends on several factors. The three most influential factors are the order, n, of the n-gram model, the smoothing technique used, and the number of training documents. These factors significantly influence the quality of a language model and thus may influence classification performance. We will first discuss the relationship between language modeling quality and classification performance. Then we will further analyze the influence of each individual factor. We will also discuss other factors such as the influence of prior information $\Pr(c)$ in Equ. (4.15) and whether character or word level models should be applied.

4.5.1 Relationship between Classification Performance and Language Modeling Quality

Figures 4.5 shows the relationship between classification performance and language modeling quality on the Greek authorship attribution task. (The other data sets have similar curves.) The upper part of the figure shows classification performance and the lower part shows language modeling quality measured by average entropy across all testing documents (bits per character). We can see that classification performance is almost monotonically related to language modeling quality. However, this is not absolutely true. Since our goal is to make a final decision based on the *ranking* of perplexities, not just their absolute values, a slightly superior language model in the sense of perplexity reduction (i.e. from the perspective of classical language modeling) does not necessarily lead to a better decision from the perspective of categorization accuracy.

The influence of language modeling can further be decomposed into three factors: or-

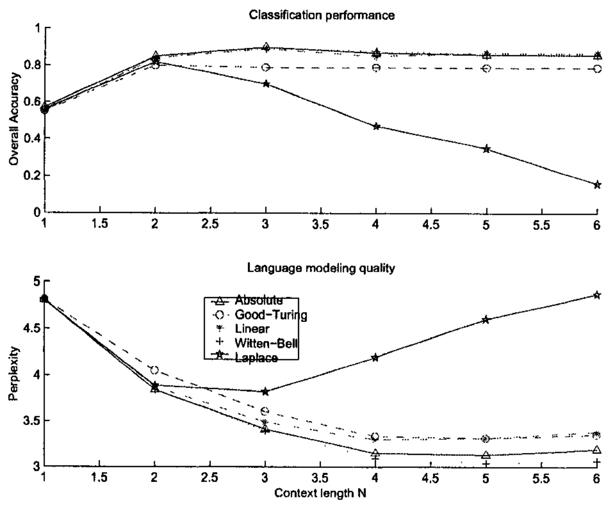

Figure 4.5: Relationship between classification performance and language modeling quality

der of the n-gram model, smoothing techniques, and number of training documents. We analyze these factors individually below.

4.5.2 Effects of n-Gram Order

Relaxing the naive Bayes independence assumption to consider local context dependencies is one of the main motivations of the CAN Bayes model. The order n is a key factor in determining the quality of n-gram language models. If n is too small then the model will not capture enough context. However, if n is too large then this will create severe sparse data problems. Both extremes result in a larger perplexity than the optimal context length and decrease classification performance. Figures 4.6 illustrates the influence of order n on classification performance in the previous five experiments (Greek authorship attribution, Greek text genre classification, English topic detection, Chinese topic detection and Japanese topic detection) using absolute smoothing. From the curves, one can see that

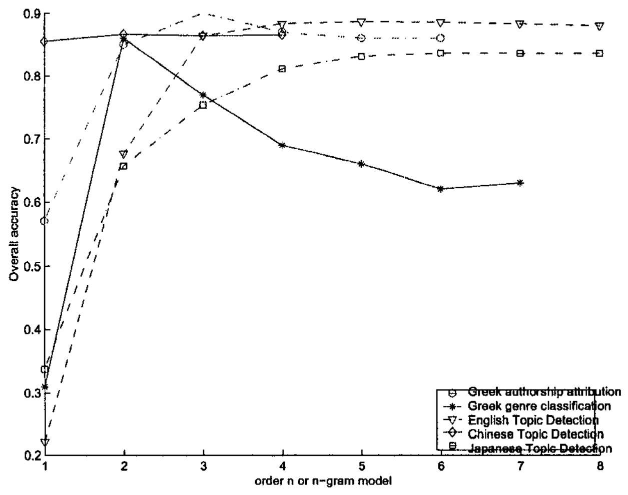

Figure 4.6: Influence of the order n on classification performance

as the order increases, classification accuracy increases, presumably because the longer context better captures the regularities of the text. However, at some point accuracy begins to decrease and entropy begins to increase as sparse data problems begin to set in. Interestingly, the effect is more pronounced in some experiments (Greek genre classification) but less so in other experiments (topic detection under any language). The sensitivity demonstrated in the Greek genre case could still be attributed to the sparse data problem (over-fitting in genre classification could be more serious than other problems).

4.5.3 Effects of Smoothing Technique

Another key factor affecting the performance of a language model is the smoothing technique used. We illustrate the results on the 20 Newsgroup data sets with character level and word level models in Figure 4.7 and Figure 4.8 respectively.

Here we find that, Laplace smoothing over-fits very quickly. Other smoothing tech-

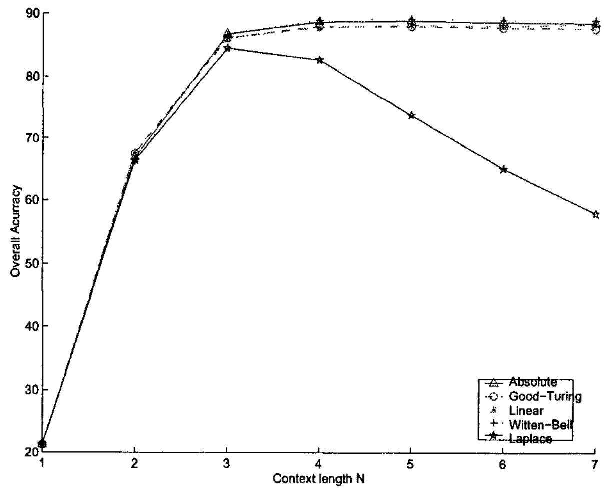

Figure 4.7: Results on the 20 Newsgroup data using character models

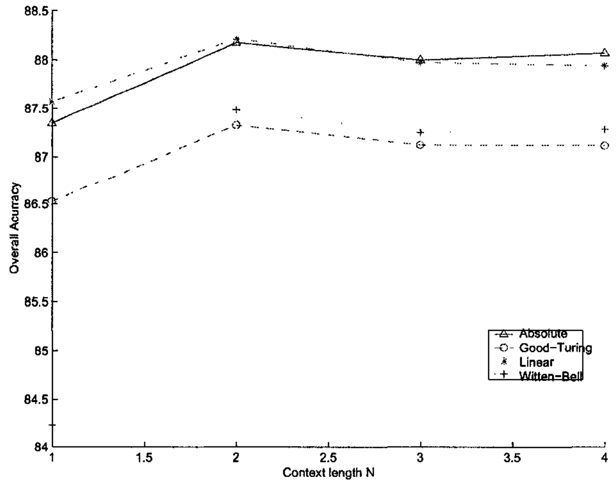

Figure 4.8: Results on the 20 Newsgroup data using word models (Laplace smoothing is not shown for clarity)

niques perform similarly on character models but vary significantly on word level models. On word level models, absolute smoothing and linear smoothing outperform other smoothing techniques. However, in character level models, for the most part, one can use any standard smoothing technique (except Laplace smoothing) and obtain comparable performance. One reason that the smoothing technique does not make a big difference for character level models is that character level models have a very small vocabulary.

4.5.4 Influence of Training Size

Clearly, the size of the training corpus can affect the quality of a language model. Normally with a larger training corpus more reliable statistics can be recovered which leads to better prediction accuracy on test data. To test the effect of training set size we obtained an additional 10 documents from each author in the group B Greek data set. In fact, this same additional data has been used in [136] to improve the accuracy of their method from 72% to 87%. Here we find that the extra training data also improves the accuracy of our method, although not so dramatically.

Figure 4.9 shows the improvement obtained for n-gram language models using absolute smoothing. Here we can see that indeed extra training data uniformly improves attribution accuracy. On the augmented training data the best model (tri-gram) now obtains a **92%** attribution accuracy, compared to the 90% we obtained originally. Moreover, this improves the best result of 87% obtained earlier [136]. Our smaller relative improvement could be due to the fact that it is harder to reduce a small prediction error. A similar phenomenon also occurs with other smoothing techniques.

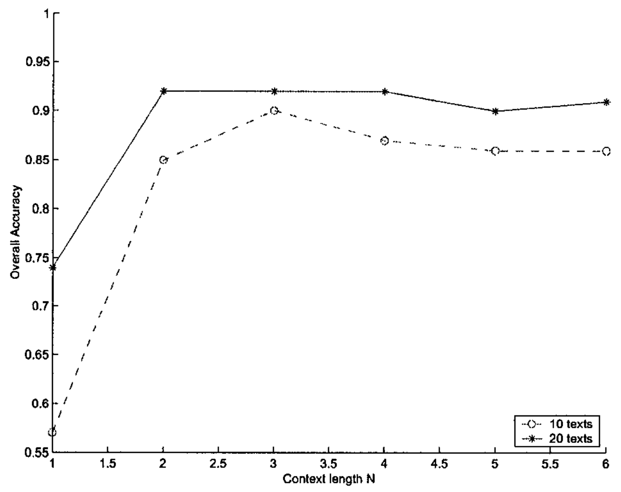

Figure 4.9: Influence of training size on accuracy.

Task	Character level	Word level
Greek authorship attribution	90%	96%
Greek genre detection	86%	81%
English topic detection (20news)	89.1%	88%
English sentimental classification	73%	81.5%

Table 4.19: Character level versus word level results

4.5.5 Character Level versus Word Level

For many Asian languages such as Chinese and Japanese, where word segmentation is hard, character level CAN Bayes models are well suited for text classification because they avoid the need for word segmentation. For Western languages such as Greek and English, one can work at both the word and character levels. Table 4.19 compares character and word level results on the Greek and English experiments.

In the Greek authorship attribution task and movie review sentimental classification task, word level models significantly outperform character level models. However, in other experiments, character level models outperform word level models. It seems that for informal texts, such as Newsgroup data and other on-line data, character models have an advantage since that they can capture some regularities that word level models miss in this case, such as spelling errors. By relaxing the context, character level models can also capture regularities at the word level, and even phrase level regularities. However, it is not clear whether a character or word model should be used for a specific text. Perhaps combining the two levels could result in more robust and more accurate results.

4.5.6 Choice of Priors

The prior $\Pr(c)$ in Equ.(4.15) can be empirically computed from training data, or sometimes can be assumed to be uniform if we do not know its true distribution. In the Japanese topic detection experiments, the data had 24 categories and documents were distributed unevenly between categories (with a minimum of 1747 and maximum of 53975 documents per category). This imbalanced distribution could cause some difficulty if one assumed a uniform prior over categories. However, with an empirical prior distribution learned from training data, we observe that the results do not change significantly from simply using a uniform prior. The comparison of using a uniform prior versus an empirical prior is shown in Table 4.20 (with absolute smoothing). There is a significant improvement for uni-gram models when using an empirical instead of uniform prior. However, the difference becomes smaller and eventually can be ignored as higher order models are used. Basically, it is safe to use a uniform prior in text classification tasks.

n	Uniform prior	Empirical prior
1	0.33	0.38
2	0.66	0.67
3	0.75	0.76
4	0.81	0.81
5	0.83	0.83
6	0.84	0.84
7	0.84	0.84
8	0.83	0.83

Table 4.20: Uniform prior or empirical prior on Japanese topic detection results

4.5.7 Overall Performance Compared to State of the art

The results we have reported here are comparable to (or much better than) the state of the art results on the same data sets. For example, our 90% accuracy for Greek authorship attribution is much better than the 72% reported earlier. Our 86% Greek genre classification accuracy is better than the 81%, which is based on a much more complicated analysis. Our 89.13% accuracy on the 20 Newsgroups data set is better than the best result, 87.5%, reported in [115] which is based on a combination of SVMs and error correcting output coding (ECOC). Our 86.7% accuracy on the Chinese TREC data is better than the 81.7% achieved by SVMs. Overall, the chain augmented naive Bayes classifier works very well, even though it is a much simpler technique than these other methods and it is not specialized to any particular data set.

However, language modeling based approach does not win in all cases. We also found that in Reuters-21578 data, SVMs still outperform language modeling based classifiers. The reason is that the sparse data problem in this data set is very severe in some categories (such as corn, ship, and wheat). In these categories, a traditional SVM classifier with an explicit feature selection may have an advantage since the feature selection procedure may be able to pick up a few words which are enough to distinguish the categories, while language modeling approach does not have enough data to obtain reliable probability estimates. Nevertheless, language modeling based classifiers still improve plain naive Bayes classifiers in this case.

4.6 Relation to Previous Research

In principle, any language model can be used to perform text categorization based on Equ. (4.14). However, n-gram models are extremely simple and have been found to be effective in many applications. For example, character level n-gram language models can be easily applied to any language, and even non-language sequences such as DNA and music. Character level n-gram models are widely used in text compression—e.g., the PPM model [7]—and have recently been found to be effective in text classification problems as well [138]. The PPM model is a weighted linear interpolation n-gram models and has been set as a benchmark in text compression for decades. Building an adaptive PPM model is expensive however [7], and our back-off models are relatively much simpler. Using compression techniques for text categorization has also been investigated elsewhere [9], where the authors seek a model that yields the minimum compression rate increase when a new test document is introduced. However, this method is found not to be generally effective nor efficient [53]. In our approach, we evaluate the perplexity (or entropy) directly on test documents, and find the outcome to be both effective and efficient.

4.7 Summary

We have presented a simple approach for language and task independent text categorization based on character level n-gram language modeling. The approach is evaluated on four different languages and five different text categorization problems. Surprisingly, we observe state of the art or better performance in most cases. We have also experimentally analyzed the influence of different factors that can affect the accuracy of this approach, and found that for the most part the results are robust to perturbations of the basic method. The wide applicability and simplicity of this approach makes it immediately applicable to any sequential data (such as natural language, music, DNA) and yields effective baseline performance. The success of this simple method, we think, is due to the effectiveness of well known statistical language modeling techniques, which surprisingly have had little significant impact on the learning algorithms normally applied to text categorization. Nevertheless, statistical language modeling is also concerned with modeling the semantic, syntactic, lexicographical and phonological regularities of natural language—and would seem to provide a natural foundation for text categorization problems. One interesting difference, however, is that instead of explicitly pre-computing features and selecting a subset based on arbitrary decisions, the language modeling approach simply considers all character (or word) subsequences occurring in the text as candidate features, and implicitly considers the contribution of *every* feature in the final model. Thus, the language modeling approach completely avoids a potentially error-prone feature selection process. Also, by applying character-level language models, one also avoids the word segmentation problems that arise in many Asian languages, and thereby achieves a language independent method for constructing accurate text categorizers.

To us, these results suggest that basic statistical language modeling ideas might be more relevant to other areas of natural language processing than commonly perceived.

Chapter 5

Language Independent Unsupervised Lexical Learning and Word Segmentation

In this chapter, we present a machine learning approach for automated lexical learning based on the EM algorithm and n-gram models. The approach is language independent and completely unsupervised. Thus it removes the necessity of manual lexicon construction and can be easily adapted to other languages. Word segmentation is accomplished by a dynamic programming algorithm (Viterbi decoding), which segments sequences dynamically based on their context. We propose several methods to reduce the data sparse problem and allow EM to escape local maxima. The proposed augmentations include mutual information based lexicon pruning, a hierarchical approach to employing word structure information, and a self-supervised method for lexicon construction.

5.1 Introduction

Word segmentation is an important problem in many natural language processing tasks. For example, a first step in speech recognition is to segment the continuous speech utterance into word-like units for further processing. In many written Asian languages like Chinese, Japanese, and Thai, word segmentation is also a problem. Unlike English and other western languages where words are explicitly delimited by white-spaces, these Asian language do not have any delimiters between words. Word segmentation is therefore a key sub-problem of many language processing tasks (such as information retrieval and machine translation) in these languages. Beyond natural language processing tasks, similar segmentation problems have also recently arisen in bio-informatics where continuous protein sequences need to be segmented into meaningful patterns.

Unfortunately, segmenting an input sentence into words is a nontrivial task. There has been a significant amount of research on techniques for discovering word segmentation boundaries; see for example [2, 3, 14, 13, 21, 27, 28, 36, 37, 49, 60, 69, 75, 76, 112, 133, 134, 154], among which there are at least two Ph.D. theses [36, 75]. The main idea behind most of these techniques is to start with a lexicon that contains the set of possible words and then segment a concatenated character string by optimizing a heuristic objective such as maximum length match, mutual information, or maximum likelihood. This approach implies, however, that one of the main problems in word segmentation is constructing the original lexicon.

Lexicon construction methods can be classified as either supervised or unsupervised. In supervised lexicon construction, one has to segment a raw unsegmented corpus by hand and then collect all the words from the segmented corpus to build a lexicon. However, the supervised learning method for constructing a lexicon has received much criticism because of its obvious disadvantages: First, the lexicon constructed by this method cannot be complete. Second, the supervised lexicon construction method is not adaptive. That is, the lexicon constructed in one domain is not applicable to another domain. Third, it requires too much human labor to segment a large corpus manually.

Therefore many unsupervised methods have been proposed for segmenting raw character sequences into words with no boundary information. Brent [13] gives a good survey of the area. Most current approaches are based on using some form of EM (expectation-maximization) to learn a probabilistic speech or text model and then employing Viterbi-decoding-like procedures to segment new speech or text into words. Another unsupervised method that has been used in Chinese text retrieval is mutual information based segmentation [24]. This method limits words to be of length of most two characters and thus is not a general word segmentation method. In this chapter, we focus on EM based approaches only.

One reason that the EM algorithm is widely adopted for unsupervised learning is that it is guaranteed to converge to a good probability model that locally maximizes the likelihood or posterior probability of the training data [38]. For the problem of word segmentation, EM is typically applied by first extracting a set of candidate n-grams from a given training corpus [37], initializing a probability distribution over this set, and then using the standard iteration to adjust the probabilities of the n-grams to increase the posterior probability of the training data.

There are at least three problems with the standard EM approach. First, because likelihood is usually defined by a product of individual chunk probabilities (making the standard assumption that segments are independent), the more chunks a segmentation has, the smaller its likelihood will tend to be. For example, given a character sequence *sizeofthecity* and a uniform distribution over n-grams, the segmentation *sizeof|thecity* will have higher likelihood than segmentation *size|of|the|city*. Therefore, the maximum likelihood EM training procedure will prefer fewer chunks in its segmentation, and thus it will tend to put a large probability on long non-word character sequences such as *sizeof, longtime* and *computerscience*. If one can break such excessive agglomerations into short legal words, such as *size, of, long, time, computer* and *science*, then the lexicon will be much smaller and both training and segmentation should be improved.

Second, in continuous speech segmentation, there is a significant sparse data problem in training large n-gram models. For example, if one wished to model all English words of length up to 15 characters, $26 + 26^2 + \cdots + 26^{15}$ n-grams would have to be considered (in a naive model that only considers lower case characters). Although EM will give zero probability to any unseen n-gram—and therefore eliminate most of them—it still typically produces a very large lexicon. Due to the limited amount of training data it remains difficult to estimate a probability distribution that is defined by so many free parameters.

Third, EM is known to have problems with getting trapped in poor local maxima [38] and often achieves results that depend strongly on the distribution from which it is initialized.

In this chapter, we propose several methods that can mitigate all three of these problems. In Section 5.4, we proposal a mutual information based lexicon pruning scheme to address the three problems. We do this by detecting long n-grams that can be decomposed into shorter n-grams without significantly reducing data likelihood, and delete these weakly connected long sequences from the lexicon. The rationale for lexicon pruning is not merely to cope with sparse data, but also to reduce excessive word agglomerations (the second problem) and help guide EM out of poor local minima (the third problem). As pointed out by Brand [12], effective parameter pruning can help move EM into better subspaces and enable further progress.

In Section 5.5, we then propose a two-level hierarchical EM method to segment words. The idea is based on the simple intuition that words themselves are built out of an intermediate vocabulary of morphemes (or phonemes in speech) which are in turn defined by shorter structured character strings. For example, in English, words (like *carelessly*) are composed of one to three morphemes, (e.g., *care, less, ly*), which are themselves composed of one to five characters. To exploit this feature of natural text, we apply EM hierarchically in two phases: first to learn a morpheme lexicon, and second to learn a word lexicon over the base morpheme vocabulary. This hierarchical approach dramatically reduces the number of free parameters in our probability model by reducing the number of candidate n-grams to $26 + 26^2 + \cdots + 26^5 + |G| + |G|^2 + |G|^3$ (where $|G|$ is the number of morphemes retained in our model)—which substantially reduces the problem of sparse training data. Although the idea of using morphemes/phonemes to detect word boundaries is not new [14, 13, 28], previous work all assumes that the set of morphemes/phonemes is fixed beforehand and therefore learns a word model over an established morpheme/phoneme vocabulary. Our work is different in that we automatically learn the underlying morpheme/phoneme vocabulary from a training set of unsegmented character/phone strings.

Finally, in Section 5.6, we propose a variant of the EM algorithm for segmenting Chinese words, which is termed self-supervised learning because this approach can automatically identify the words in a core lexicon. The idea is based on a simple observation: in the word segmentation problem, one can use known words to guide the recognition of unknown words. For example, if one knows the word "*computer*" then upon seeing "*computerscience*" it is natural to segment "*science*" as a new word. Based on this, we propose a new unsupervised training method for acquiring probability models that accurately segment Chinese character sequences into words. By automatically constructing a core lexicon to guide unsupervised word learning, self-supervised segmentation overcomes the local maxima problems that hamper standard EM training. Our procedure uses successive EM phases to learn a good probability model over character strings, and then prunes this model with a mutual information selection criterion to obtain a more accurate word lexicon. The segmentations produced by these models are more accurate than those produced by training with EM alone.

In the remainder of the chapter, we first introduce the EM algorithm and then show how EM can be used for unsupervised lexicon learning. Then we proceed to introduce mutual information based lexicon pruning, the hierarchical EM segmentation algorithm, and the self-supervised Chinese word segmentation algorithm.

5.2 Learning from Unlabeled Data with EM

In many applications, labeled training data is not easy to obtain for supervised learning, while unlabeled data is abundant (e.g., on the Web). Unsupervised learning that exploits unlabeled data for training is attracting increasing interest. Although research on exploiting unlabeled data is still maturing, it has been receiving more attention recently. Approaches to using unlabeled data for supervised learning include: expectation maximization (EM) [38], co-training [11], adaptive regularization [125], active learning [141], and transductive learning. EM, in particular, is widely used in density estimation from unlabeled data in probabilistic models.

5.2.1 Problem Formulation

EM is a general method for finding locally maximum-likelihood estimates of the parameters of an underlying distribution from data that is incomplete or has missing values.

Let us assume that a random variable X is observed and is independently identically generated by some process parameterized by Θ. We call X the *incomplete data* and assume that a complete data tuple consists of $Z = (X, Y)$ where Y is the missing component. Assuming there are t observations $X = \{x_1, x_2, ..., x_t\}$, then the goal of maximum likelihood density estimation is to find an optimal Θ^* which gives the largest likelihood $\Pr(X|\Theta)$, or log-likelihood $L(X|\Theta) = \log \Pr(X|\Theta)$, for the observation X.

$$\begin{aligned} \Theta^* &= \arg\max_{\Theta} \; L(X|\Theta) \\ &= \arg\max_{\Theta} \; \log \Pr(X|\Theta) \\ &= \arg\max_{\Theta} \; \sum_{j=1}^{t} \log \Pr(x_j|\Theta) \\ &= \arg\max_{\Theta} \; \sum_{j=1}^{t} \log \sum_{y_j \in Y} \Pr(x_j, y_j|\Theta) \end{aligned} \tag{5.1}$$

Because the objective function (5.1) contains the log of a sum, there is no general closed-form solution for Θ^*. The EM algorithm is used in this situation.

There are two steps in the EM algorithm: the expectation (E) step and the maximization (M) step. In the E step, one computes the expectation of the log-likelihood for the complete data $L(X, Y|\Theta)$ given the observed data X and current parameter estimate Θ^{k-1}. This expectation function is called the Q function.

$$Q(\Theta, \Theta^{k-1}) = E\left[\log \Pr(Z|\Theta)|X, \Theta^{k-1}\right] \tag{5.2}$$

$$= E\left[\sum_{j=1}^{t} \log \Pr(x_j, y_j|\Theta) \middle| X, \Theta^{k-1}\right] \tag{5.3}$$

The M step consists of finding a Θ which maximizes $Q(\Theta, \Theta^{k-1})$ and setting it to be the new parameters, i.e.

$$\Theta^k = \arg\max_{\Theta} \ Q(\Theta, \Theta^{k-1}) \tag{5.4}$$

The E step and M step are iterated until convergence or some stopping criterion is met.

Sometimes, if the M step is too difficult to compute, it can be replaced by a generalized M step. For a generalized M step, it is not necessary to find a Θ which maximizes the log likelihood, but to simply find a Θ which only improves the log likelihood compared to the previous step. This version of EM, called Generalized EM (GEM), converges more slowly than standard EM.

5.2.2 Convergence of EM

The proof of convergence of EM can be found in many papers [38, 32, 94]. To be self-contained, we summarize a simple proof below.

To prove the convergence of EM, we need to show each iteration will increase log-likelihood $L(X|\Theta)$. Note that the log-likelihood $L(X|\Theta)$ can be decomposed as follows.

$$L(X|\Theta) = \sum_{j=1}^{t} \log \Pr(x_j|\Theta) \tag{5.5}$$

$$= \sum_{j=1}^{t} \log \Pr(x_j, y_j|\Theta) - \sum_{j=1}^{t} \log \Pr(y_j|x_j, \Theta) \tag{5.6}$$

since $\Pr(x_j, y_j|\Theta) = \Pr(x_j|\Theta)\Pr(y_j|x_j, \Theta)$ for all y_j.

Taking the expectation of both sides, given the observed data X and current parameter estimate Θ^{k-1}, we have

$$E\left[L(X|\Theta)|\, X, \Theta^{k-1}\right] = E\left[L(Z|\Theta)|\, X, \Theta^{k-1}\right] - E\left[L(Y|X, \Theta)|\, X, \Theta^{k-1}\right] \tag{5.7}$$

$$= Q(\Theta, \Theta^{k-1}) - H(\Theta, \Theta^{k-1}) \tag{5.8}$$

where $H(\Theta, \Theta^{k-1}) = E\left[L(Y|X, \Theta)|\, X, \Theta^{k-1}\right]$.

First consider the left side

$$E\left[L(X|\Theta)|\, X, \Theta^{k-1}\right] = E\left[\sum_{j=1}^{t} \log \Pr(x_j|\Theta) \middle|\, X, \Theta^{k-1}\right] \tag{5.9}$$

$$= \int \sum_{j=1}^{t} \log \Pr(x_j|\Theta) \prod_{l=1}^{t} \Pr(y_l|x_l, \Theta^{k-1}) dy_1 ... dy_t \tag{5.10}$$

$$= \sum_{j=1}^{t} \log \Pr(x_j|\Theta) \int \prod_{l=1}^{t} \log \Pr(y_l|x_l, \Theta^{k-1}) dy_1 ... dy_t \tag{5.11}$$

$$= \sum_{j=1}^{t} \log \Pr(x_j|\Theta) = L(X|\Theta) \tag{5.12}$$

Thus

$$L(X|\Theta) = Q(\Theta, \Theta^{k-1}) - H(\Theta, \Theta^{k-1}) \tag{5.13}$$

If $H(\Theta, \Theta^{k-1}) \leq H(\Theta^{k-1}, \Theta^{k-1})$, the convergence of EM would follow straightforwardly, since this would imply

$$Q(\Theta, \Theta^{k-1}) > Q(\Theta^{k-1}, \Theta^{k-1}) \Rightarrow L(X|\Theta) > L(X|\Theta^{k-1}) \tag{5.14}$$

Theorem: $H(\Theta, \Theta^{k-1}) \leq H(\Theta^{k-1}, \Theta^{k-1})$

Proof:

We first decompose $H(\Theta, \Theta^{k-1})$ as follows.

$$\begin{aligned} H(\Theta, \Theta^{k-1}) &= E\left[L(Y|X, \Theta) | X, \Theta^{k-1}\right] & (5.15)\\ &= E\left[\sum_{j=1}^{t} \log \Pr(y_j|x_j, \Theta) \,\middle|\, X, \Theta^{k-1}\right] & (5.16)\\ &= \int \sum_{j=1}^{t} \log \Pr(y_j|x_j, \Theta) \prod_{l=1}^{t} \Pr(y_l|x_l, \Theta^{k-1}) dy_1 ... dy_t & (5.17)\\ &= \sum_{j=1}^{t} \int \log \Pr(y_j|x_j, \Theta) \Pr(y_j|x_j, \Theta^{k-1}) dy_j \prod_{l \neq j} \int \Pr(y_l|x_l, \tilde{\Theta}) dy_l & (5.18)\\ &= \sum_{j=1}^{t} \int \log \Pr(y_j|x_j, \Theta) \Pr(y_j|x_j, \Theta^{k-1}) dy_j & (5.19)\\ &= \sum_{j=1}^{t} h_j(\Theta, \Theta^{k-1}) & (5.20) \end{aligned}$$

where $h_j(\Theta, \Theta^{k-1}) = \int \log \Pr(y_j|x_j, \Theta) P(y_j|x_j, \Theta^{k-1}) dy_j$. Thus $H(\Theta, \Theta^{k-1})$ can be decomposed into a sum of a function of individual observations. We now prove

$$h_j(\Theta, \Theta^{k-1}) \le h_j(\Theta^{k-1}, \Theta^{k-1}) \quad \text{for all } j.$$

First notice that

$$h_j(\Theta, \Theta^{k-1}) = \int \log \Pr(y_j|x_j, \Theta) \Pr(y_j|x_j, \Theta^{k-1}) dy_j \tag{5.21}$$

$$h_j(\Theta^{k-1}, \Theta^{k-1}) = \int \log \Pr(y_j|x_j, \Theta^{k-1}) \Pr(y_j|x_j, \Theta^{k-1}) dy_j \tag{5.22}$$

Thus,

$$\begin{aligned} & h_j(\Theta, \Theta^{k-1}) - h_j(\Theta^{k-1}, \Theta^{k-1}) \\ = & \int \left[\log \Pr(y_j|x_j, \Theta) - \log \Pr(y_j|x_j, \Theta^{k-1})\right] \Pr(y_j|x_j, \Theta^{k-1}) dy_j & (5.23) \\ = & \int \left[\log \frac{\Pr(y_j|x_j, \Theta)}{\Pr(y_j|x_j, \Theta^{k-1})}\right] \Pr(y_j|x_j, \Theta^{k-1}) dy_j & (5.24) \end{aligned}$$

Note that since $\log x \le (x-1)$, we have

$$\log \frac{\Pr(y_j|x_j, \Theta)}{\Pr(y_j|x_j, \Theta^{k-1})} \le \frac{\Pr(y_j|x_j, \Theta)}{\Pr(y_j|x_j, \Theta^{k-1})} - 1$$

Thus,

$$h_j(\Theta, \Theta^{k-1}) - h_j(\Theta^{k-1}, \Theta^{k-1}) \le \int \left[\frac{\Pr(y_j|x_j, \Theta)}{\Pr(y_j|x_j, \Theta^{k-1})} - 1\right] \Pr(y_j|x_j, \Theta^{k-1}) dy_j \tag{5.25}$$

which implies

$$h_j(\Theta, \Theta^{k-1}) - h_j(\tilde{\Theta}, \Theta^{k-1}) \leq \int \left[\Pr(y_j|x_j, \Theta) - \Pr(y_j|x_j, \Theta^{k-1})\right] dy_j = 0 \tag{5.26}$$

and hence

$$h_j(\Theta, \Theta^{k-1}) \leq h_j(\Theta^{k-1}, \Theta^{k-1}) \tag{5.27}$$

Combining Equ. (5.13), Equ. (5.20), and Equ. (5.27) establishes Equ. (5.14). Thus the convergence of EM is proven.

5.2.3 Comments on Applying EM

Since EM only finds local maxima, applying EM directly often does not yield good performance. Some variants have been used in various applications. For example, McCallum and Nigam [91] use an active learning method to select unlabeled data in a text classification problem. To overcome the local maximum problem of standard EM, the authors initialize EM only with labeled data, and then use a criterion derived from *Query-By-Committee* (QBC) [128] to select a few most *informative* unlabeled examples to add to the labeled data pool, and then rerun EM. Nigam et al. [101] present a case study which addresses whether unlabeled data can help improve a Naive Bayes text classifier with only a small amount of labeled training data. To penalize the influence of unlabeled data, the authors multiply a constant λ, $0 \geq \lambda \leq 1$, with the log likelihood term of the unlabeled data. In [107], the distribution is separated into two equal parts to emphasize the influence of a set of core lexicon words. Wu et al. [150] combines EM with multiple discriminant analysis [41] and proposed a discriminant EM (D-EM) algorithm for an image retrieval application. Ivanov et al. [66] modified the EM algorithm for reinforcement learning by including the reward in the optimization problem.

From the literature and our experience, we suggest that to make EM work one has to address the following issues:

1. The initialization parameters must usually be chosen carefully (e.g. with labeled data) instead of randomly.
2. Often EM must be customized for the specific application, e.g. [91, 107].
3. Unlabeled data often must be chosen carefully; for example by selecting representative unlabeled data using active learning.
4. The iterative procedure must be monitored carefully and stopped early if necessary. Sometimes a high likelihood does not necessarily imply good performance for the application at hand, and continued iteration may result in performance degradation. That is, too many iterations can result in over-fitting [43]. We observe this effect in Chinese word segmentation.

5.3 Standard EM Word Segmentation

Assume we have a sequence of characters $C = c_1c_2...c_T$ that we wish to segment into chunks $S = s_1s_2...s_M$, where the chunks s_i are chosen from a lexicon $V = \{s_i, i = 1, ..., |V|\}$. If we already have a probability distribution $\theta = \{\theta_i|\theta_i = p(s_i), i = 1, ..., |V|\}$ defined over the lexicon, then we can compute the most likely segmentation of the sequence $C = c_1c_2...c_T$

into chunks $S = s_1 s_2 ... s_M$ as follows. First, for any given segmentation S of C, we can calculate the joint likelihood of S and C by

$$\Pr(S, C|\theta) = \prod_{i=1}^{M} \theta_i \tag{5.28}$$

Our task is to find the segmentation S^* that achieves the maximum likelihood:

$$\begin{aligned} S^* &= \arg\max_S \Pr(S|C;\theta) \\ &= \arg\max_S \Pr(S, C|\theta) \end{aligned} \tag{5.29}$$

Note that the number of possible segmentations is 2^{T-1}, and therefore it is not feasible to enumerate all of them to determine S^*. However, the best segmentation can be efficiently recovered by the Viterbi algorithm, which employs a dynamic programming approach to build up the best segmentation in a bottom up manner.

5.3.1 Viterbi decoding

Let $\delta(t)$ denote the partial best likelihood for current observation $c_1...c_t$ and let ℓ be the length of a word, which could be from 1 to the maximum word length of words L. To compute the best segmentation S^* in Equ. (5.29), one can compute $\delta(T)$ using the following equation and track the best word length ℓ in each step

$$\delta(t) = \max_{\ell \in \{1..L\}} \delta(t-\ell) \times \Pr(c_{t-\ell+1}...c_t) \tag{5.30}$$

and $\delta(t) = 1$ when $t = 0$. In practice, one often uses log-arithmetic to avoid underflow.

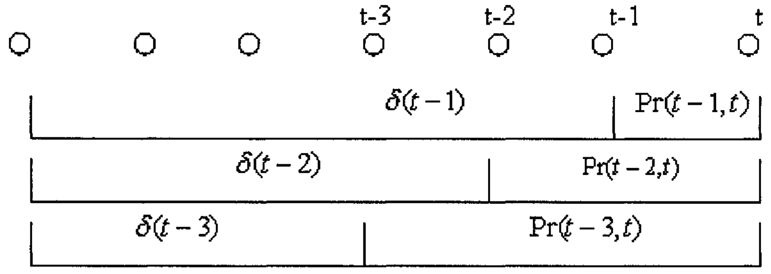

Figure 5.1: The successive computation procedure for Viterbi-decoding

The algorithm is illustrated in Figure 5.1.

Here, the time complexity is $O(LT)$ instead of 2^{T-1}. Viterbi decoding is very efficient when T is large (see Rabiner [114] for more details of the Viterbi algorithm).

5.3.2 Parameter Re-estimation

Viterbi decoding allows one to compute the best segmentation for a given probability distribution $\theta = \{p(s_i)\}$ over the lexicon. *Learning* these probabilities from a training corpus is the job of the EM algorithm. Following Dempster et al. [38], the update Q function that we use in the case we are examining here is

$$Q(k, k+1) = \sum_S \Pr(S|C; \theta^k) \log(\Pr(C, S|\theta^{k+1})) \tag{5.31}$$

By maximizing Equ. (5.31) under the constraint that $\sum_i \theta_i^{k+1} = 1$, we obtain the parameter re-estimation formula

$$\theta_i^{k+1} = \frac{\sum_S \#(s_i, S) \times \Pr(S, C|\theta^k)}{\sum_{s_j} \sum_S \#(s_j, S) \times \Pr(S, C|\theta^k)} \tag{5.32}$$

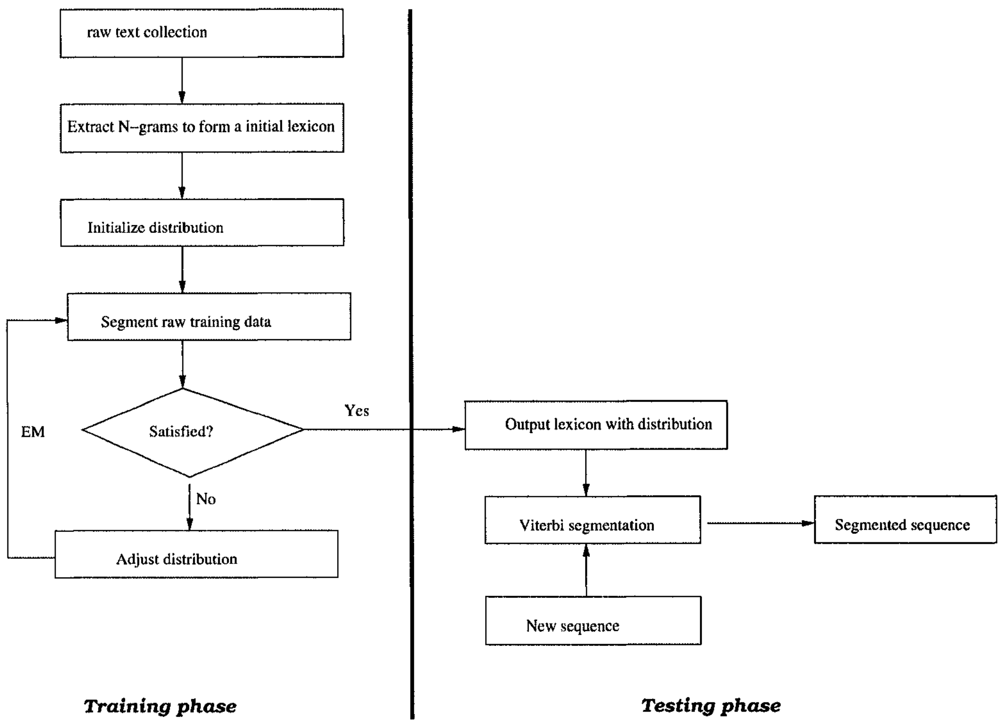

Figure 5.2: Procedures of unsupervised word segmentation

Here the numerator is a sum over all possible segmentations S of the number of occurrences of a word s_i, weighted by the probability of the segmentation. Similarly, the denominator is a weighted sum of the number of words in all possible segmentations. Thus, Equ. (5.32) is a weighted frequency count. Equ. (5.32) can be efficiently calculated by using the forward and backward algorithm, or be efficiently approximated by using the Viterbi algorithm; see [114] and [37] for detailed algorithms.

Figure 5.2 shows the procedure of unsupervised word segmentation.

5.4 Mutual Information Lexicon Pruning

A naive application of EM will run into at least two of the problems we identified in Section 5.1: excessive agglomeration and poor local minima. To cope with both of these issues, we employ a simple lexicon pruning scheme that eliminates long agglomerations of short primitives. For example, a naive application of EM results in agglomerated character sequence such as *sizeof* to be learned in the initial model and retrieved in subsequent segmentation. Clearly, the string *sizeof* is a concatenation of two shorter legal words, *size* and *of*. As mentioned previously, maximizing likelihood does not necessarily remove these larger agglomerations, and in fact encourages their creation. To prune these conjunctions of shorter words (and morphemes) we employ a probabilistic criterion based on mutual information.

Recall that the mutual information between two random variables X and Y is defined as

$$MI(X,Y) = \sum_{x,y} \Pr(X=x, Y=y) \log \frac{\Pr(X=x, Y=y)}{\Pr(X=x) \times \Pr(Y=y)} \tag{5.33}$$

where a large value indicates strong dependence and zero indicates independence. To implement our pruning criteria we use a variant of this formula to evaluate the cohesiveness of strings. Specifically, given a long string s we consider splitting it into the two substrings s_1 and s_2 that maximize $p(s_1) \times p(s_2)$ over all two-chunk segmentations $s = s_1 s_2$. Let the probabilities of the original string and the two chunks be $p(s)$, $p(s_1)$ and $p(s_2)$ respectively. We define a modified *pointwise mutual information* [87] between s_1 and s_2 to be

$$pMI(s_1, s_2) = \frac{1}{T} \times \log \frac{p(s)}{p(s_1) \times p(s_2)}. \tag{5.34}$$

where T is the length of s. To apply this measure to pruning, we set two thresholds $\gamma_1 > \gamma_2$. If the mutual information is higher than the high threshold γ_1 we say that s_1 and s_2 are strongly correlated and do not split s. (That is, we do not remove s from the lexicon.) If the mutual information is lower than the lower threshold γ_2 we say that s_1 and s_2 are nearly independent, so we remove s from the lexicon and distribute its probability to s_1 and s_2. If the mutual information is between the two thresholds we say that s_1 and s_2 are weakly correlated and therefore shift some of the probability from s to s_1 and s_2 by keeping a portion of s's probability for itself (1/3 in our experiments below) and distribute its remaining probability mass to the smaller chunks proportional to their probabilities. The idea is to shift the weight of the probability distribution toward shorter words.

This splitting process is carried out recursively for s_1 and s_2. The pseudo code is illustrated in Figure 5.3.

5.5 Hierarchical EM Word Segmentation

To cope with the remaining problem identified in Section 5.1 (the sparse data problem), we propose a two level hierarchical EM approach for unsupervised word segmentation. In the first level, we generate all morphemes with one to five characters from the training corpus C, use EM to learn a probability distribution over morphemes (as described above), prune low probability morphemes, and segment the original training corpus C into a morpheme sequence G. In the second level, we generate a large word lexicon from G (n-grams over morphemes), use EM to learn a probability distribution over words, and segment G into a word sequence W. The complete process is illustrated in Figure 5.4. In both phases, once EM converges we employ additional lexicon pruning (discussed in Section 5.4) which pulls

```
1: $(s_1, s_2) = mostlikely\_split(s)$;

2: $MI = \frac{1}{T} \times log\frac{p(s)}{p(s_1) \times p(s_2)}$

3: if($MI > \gamma_1$){//strongly dependent
     return -1;
   }
   else if($MI < \gamma_2$){//independent
     $probSum = p(s_1) + p(s_2)$;
     $p(s_1) += p(s) \times p(s_1)/probSum$;
     $p(s_2) += p(s) \times p(s_2)/probSum$;
     $p(s) = 0$;
     return 1;
   }
   else{//weakly dependent
     $probDistribute = p(s)/3$;
     $probSum = p(s_1) + p(s_2)$;
     $p(s) = probDistribute$;
     $p(s_1) += 2 \times probDistribute \times p(s_1)/probSum$;
     $p(s_2) += 2 \times probDistribute \times p(s_2)/probSum$;
     return 0;
   }
```

Figure 5.3: Mutual information probabilistic lexicon pruning

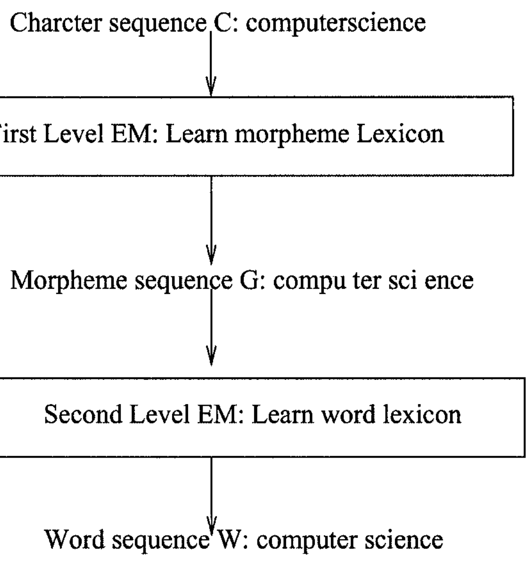
Charcter sequence C: computerscience
First Level EM: Learn morpheme Lexicon
Morpheme sequence G: compu ter sci ence
Second Level EM: Learn word lexicon
Word sequence W: computer science

Figure 5.4: Hierarchical EM segmentation model

EM out of local maxima and allows it to make further progress. Overall, the first level determines

$$G^* = \arg\max_G \Pr(G, C|\theta_g) \tag{5.35}$$

and the second level determines

$$W^* = \arg\max_W \Pr(W, G|\theta_w) \tag{5.36}$$

where θ_g and θ_w are the distributions over morpheme lexicon and word lexicon respectively. The EM algorithms in both levels are identical except that in the first level the basic observation unit is the character and in the second level the basic unit is the morpheme.

5.5.1 Performance measures

We measured performance on the test corpus by recording recall, precision and F-measures with respect to detecting the word boundaries in a test sequence. A predicted word boundary which corresponds to white space or punctuation in the original test text is considered correct [76]. Let N_b^1 denote the true number of word boundaries in the original test text, let N_b^2 denote the number of predicted boundaries, of which N_b^3 are correct. Then the precision, recall and F-measure on boundary are defined by

$$\text{boundary precision: } p_b = \frac{N_b^3}{N_b^2}$$
$$\text{boundary recall: } r_b = \frac{N_b^3}{N_b^1}$$
$$\text{boundary F-measure: } F_b = \frac{2 \times p_b \times r_b}{p_b + r_b}$$

We also report the word identification performance on the test text [13, 14, 28, 49, 112]. A word is considered to be correctly recovered if and only if [102]:

1. a boundary is correctly placed in front of the first character of the word,
2. a boundary is correctly placed at the end of the last character of the word,
3. and there is no boundary between the first and last character of the word.

Let N_w^1 denote the number of words in the test corpus, let N_w^2 denote the number of words in the recovered corpus, and let N_w^3 denote the number of words correctly recovered. Then the precision, recall and F measures on word are defined by

$$\text{word precision: } p_w = \frac{N_w^3}{N_w^2}$$
$$\text{word recall: } r_w = \frac{N_w^3}{N_w^1}$$
$$\text{word F-measure: } F_w = \frac{2 \times p_w \times r_w}{p_w + r_w}$$

5.5.2 Experimental Results

We tested our procedure on segmenting the Brown corpus of English text into words. Specifically, we converted the corpus to lowercase letters and removed all white-space and punctuation. We then split the corpus into a training sequence C_1 (4292K characters, 891524 words, 37930 unique words) and a test sequence C_2 (317K characters; 64338 words; 9506 unique words, 2280 of which do not occur in the training sequence). For pruning we used the thresholds $\gamma_1 = 3$ and $\gamma_2 = 0.5$.

Results of the first level model

To use the limited training corpus effectively, we first trained the morpheme model on the training sequence C_1 with n-grams of lengths 1–5 characters using EM, ranked the morphemes according to their probabilities, and then picked the top rank morphemes to compose a morpheme lexicon. Our final lexicon consisted of 13506 morphemes. With this morpheme lexicon, we then segmented the training sequence C_1 into a morpheme string G_1 using the Viterbi algorithm. With this initial morpheme lexicon in hand, we then conducted several stages of our recursive pruning procedure to remove large morphemes from the lexicon. For each pruned lexicon, EM was run again to re-estimate the probability parameters, and each model was then used to re-segment the training sequence.

Each of these models was tested on the separate test sequence C_2 by running Viterbi to segment it into a morpheme string G_2 using the learned models. Table 5.1 shows the results of using the various learned probability models to predict word boundaries in the test sequence C_2 (where # represents the size of the lexicon). Here, each reduced and trained model was used to segment the test sequence using Viterbi, and the morpheme boundaries

	#	p_b	r_b	F_b
G Start	13506	0.603	0.799	0.678
G Pruned 1	8274	0.595	0.867	0.695
G Pruned 2	7793	0.590	0.873	0.693
G Pruned 3	7625	0.589	0.876	0.694
G Pruned 4	6800	0.576	0.892	0.689
G Pruned 5	6758	**0.570**	**0.908**	**0.690**

Table 5.1: Test segmentation results with the first level (morpheme) model showing boundary detection scores

were used to predict word boundaries in the initial text. Clearly, this segmentation should not work particularly well because proper morphemes only comprise sub-components of words. Nevertheless, we should see a high recall score (every word boundary corresponds to a morpheme boundary) along with a mediocre precision score—which is exactly what Table 5.1 shows.

Using the best of these morpheme models (G Pruned 5), the first 10 sentences in the test sequence are segmented as follows. (The predicted morpheme boundaries are indicated by | and the true word boundaries delimited by white spaces.)

| the| ful|t|on| count|y| grand| jury| said| fri|day| an| inves|tigat|ion of| at|lanta| s| re|cent| prim|ary| elect|ion| pro|duced| no| e|vi|dence| that| any| irre|gular|ities| took| place| the| jury| furth|er| said| in ter|m en|d| pres|ent|ments| that| the| city| execu|tive| comm|ittee| which| had| over| all| charg|e| of the| elect|ion| de|serve|s| the| prais|e| and| than|k|s| of the| city| of| at|lanta| for| the| mann|er| in| which| the| elec|t|ion| was| con|duct|ed| the| sept|ember| octob|er| term| jury| had| been| charg|ed by| ful|t|on| sup|erior| court| judge| dur|wood| p|y|e| to| inves|tigat|e r|e|port|s of| pos|sible| irre|gular|ities| in the| hard| f|ought|

	#	p_b	r_b	F_b
Start	812841	0.806	0.670	0.716
W Pruned 1	278608	0.754	0.720	0.721
W Pruned 2	224619	0.743	0.731	0.721
W Pruned 3	198723	**0.746**	**0.792**	**0.754**
W Pruned 4	181316	0.733	0.806	0.753
W Pruned 5	174806	0.725	0.806	0.749

Table 5.2: Test segmentation results with the second level (word) model showing boundary detection scores

prim|ary| which| was| w|on| by| may| or| nomin|ate| i|v|an| all|en| j|r| only| a re|l|ative| hand|ful| of| such| re|ports| was| recei|ved| the| jury| said| con|sider|ing| the| wide s|pread| inter|es|t| in the| elect|ion| the n|umber| of| voter|s and| the| size| of| this| city| the| jury| said| it| did| find| that| many| of| georg|i|a s re|gistr|ation| and| elect|ion| law|s are| out| mod|ed| or| in| adequ|ate| and| often| ambi|guous|

Results of the second level model

Using the best morpheme model learned in the first level (G Pruned 5), we then trained the second level (word) model over the morpheme segmentation of the training sequence. Using EM we learn a lexicon of words composed of morpheme strings and a probability distribution over these words. As above, we successively prune the word lexicon and re-estimate the probability distribution over words in the training sequence using EM. For each successive word model, we segment the test morpheme string into words using Viterbi decoding. The results are shown in Table 5.2. Here, the effects of pruning have an opposite effect on precision and recall.

In addition to these boundary detection results we also measured the word detection performance of the best model (W pruned 3) on the test sequence, obtaining **49.1%** word precision, **60.1%** word recall and **53.8%** F-measure. Using this best word model (W Pruned 3), the first 10 sentences in the test sequence are segmented as follows.

| *the*| *fulton*| *county*| *grand*| *jury*| *said*| *friday*| *an*| *investigat*|*ion*| *of*| *atlanta*| *s*| *recent*| *primary*| *election*| *produced*| *no*| *evidence*| *that*| *any*| *irregularities*| *took*| *place*| *the*| *jury*| *further*| *said*| *in*| *term*| *end*| *present*|*ment*|*s*| *that*| *the*| *city*| *executive*| *committee*| *which*| *had*| *over*| *all*| *charge*| *of the*| *election*| *deserve*|*s*| *the*| *praise*| *and*| *thank*|*s*| *of the*| *city*| *of*| *atlanta*| *for*| *the*| *manner*| *in*| *which*| *the*| *election*| *was*| *conduct*|*ed*| *the*| *september*| *octob*|*er*| *term*| *jury*| *had been*| *charg*|*ed by*| *fulton*| *superi*|*or*| *court*| *judge*| *dur*|*wood*| *p*|*y*|*e*| *to*| *investigat*|*e* *r*|*e*|*port*|*s of*| *possible*| *irregularities*| *in*| *the*| *hard*| *fought*| *primary*| *which*| *was*| *won*| *by*| *may*| *or*| *nominate*| *i*|*v*|*an*| *all*|*en*| *j*|*r*| *only*| *a*| *relative*| *handful*| *of*| *such*| *report*|*s*| *was*| *receiv*|*ed*| *the*| *jury*| *said*| *consider*|*ing*| *the*| *wide*| *spread*| *interest*| *in*| *the*| *election*| *the*| *number*| *of*| *voter*|*s*| *and*| *the*| *size of*| *this*| *city*| *the*| *jury*| *said*| *it*| *did*| *find*| *that*| *many*| *of*| *georgi*|*a s* *re*|*gistr*|*ation*| *and*| *election*| *law*|*s are*| *out*| *mod*|*ed*| *or*| *in*| *adequate*| *and*| *often*| *ambiguous*|

Results of a flat model

To verify the effectiveness of our hierarchical approach, we re-ran the experiments using a basic flat model that is similar to the first level of our hierarchical model. Here, we generated all words up to 15 characters from the training sequence C_1, ran EM to learn a probability model over words, and tested the model by segmenting the test sequence C_2. The results for the original trained model as well as successively pruned versions are shown in Table 5.3.

Using this flat one level model, we obtained a best word detection performance (F Pruned 1) of: **32.4%** word precision, **48.1%** word recall and **38.1%** F-measure, which is substantially below that obtained by the hierarchical model.

	#	p_b	r_b	F_b
Start	8578589	0.727	0.556	0.607
F Pruned 1	1316725	**0.846**	**0.513**	**0.620**
F Pruned 2	1232529	0.713	0.515	0.583

Table 5.3: Test segmentation results with the flat model showing boundary detection scores

5.5.3 Interpreting the results

From both Table 5.1 and Table 5.2, it is clear that pruning greatly reduces the size of lexicon and improves overall performance. In the first level model pruning increases recall from 0.799 to 0.908, which in fact makes the positive results of the second level possible. Since the second level model only combines adjacent morphemes, the second level recall must be no more than the first level recall. However, the precision is substantially higher, as hoped. As lexicon pruning continues, long n-grams are successively dropped. Consequently, the algorithm produces more chunks in the segmentation, which increases recall, but decreases precision. However, the trade-off in the F-measure usually increases at first, but then drops off when excessive pruning is performed.

The weakest results we obtained with the hierarchical model are those with no lexicon pruning: 80.6% precision, 67% recall and 71.6% F-measure. The best results were achieved after applying pruning to both the morpheme and word models. The combination G-Pruned-5 and W-Pruned-3 yielded 74.6% precision, 79.2% recall and F-measure 75.4%.

Compared to the flat model, the hierarchical model gains **13.4%** improvement on boundary detection F-measure and **15.7%** on word detection F-measure. The best known results on segmenting the Brown corpus that we are aware of are due to Kit and Wilks [76] who use a description length gain method. They trained their model on the whole corpus (6.13M) and reported results on the *training* set, obtaining a boundary precision of 79.33% and a boundary recall of 63.01% (they did not report boundary F-measure, but we can calculate it to be 70.23% in this case). By comparison, we train our model on a much smaller subset of the corpus (4292K) and test on *unseen* data. Even the weakest (un-pruned) results of the hierarchical model are better than those reported in [76]. After the lexicon is optimized, we obtain 16.19% higher recall and 4.73% lower precision; resulting in an improvement of **5.2%** in boundary F-measure.

De Marcken [36] also uses a minimum description length (MDL) framework and a hierarchical model to learn a word lexicon from raw speech. However, this work does not explicitly yield word boundaries, but instead recursively decomposes an input string down to the level of individual characters. As pointed out by Brent [13], this study gives credit for detecting a word if any node in the hierarchical decomposition spans the word. Under this measure [36] reports a word recall rate of 90.5% on the Brown corpus. However, his method creates numerous chunks and therefore only achieves a word precision rate of 17%.

Christianson et al. [28] use a simple recurrent neural network approach and report a word precision rate of 42.7% and word recall rate of 44.9% on spontaneous child-directed British English.

Brent and Cartwright [14] use an MDL approach and report a word precision rate of 41.3% and a word recall rate of 47.3% on the CHILDES collection. More recently, Brent [13] achieves improved results (about 70% word precision and 70% word recall) by employing additional language modeling and smoothing techniques.

The best word recognition performance we obtain is 49.1% word precision and 60.1% word recall, hence 53.8% word F-measure on the Brown corpus. This is better than [28, 14] but worse than [13]. However, it is difficult to draw a direct comparison between these results because of the different test corpora used. Nevertheless, our results seem to support the utility of exploiting a simple hierarchical model for word recognition.

5.6 Self-supervised Chinese Word Segmentation

Unlike English, Chinese words do not have delimiters between them. For example, given the Chinese character string "加拿大滑铁卢大学计算机系" if one does not understand Chinese, one can not tell where to put the whitespaces to segment this string, athough it is trival for Chinese speakers: "加拿大(Canada) 滑铁卢(Waterloo) 大学(University) 计算机(Computer) 系(Department)"

In the previous section, we have described a hierarchial approach to improving standard EM word segmentation. This approach is also useful in detecting compounds and phrases in English, but this method is not useful in Chinese word segmentation where most Chinese words are only 1 to 4 characters long. In this section, we describe another improvement to standard EM word segmentation which is not only useful for English, but also useful for Chinese.

One advantage of unsupervised lexicon construction is that it can automatically discover new words once other words have acquired high probability [34]. For example, if one knows the word "*computer*" then upon seeing "*computerscience*" it is natural to segment "*science*" as a new word. Based on this observation, we propose a new word discovery method that is a variant of standard EM training, but avoids getting trapped in local maxima by keeping two lexicons: a *core* lexicon that contains words that are judged to be familiar, and a *candidate* lexicon which contains all other words that are not in the core lexicon. We use EM to maximize the likelihood of the training corpus given the two lexicons, which automatically suggests new words as candidates for the core. However, once new words have been added to the core, EM is re-initialized by giving a certain fixed amount of the total probability mass to the core lexicon, thus allowing core words to guide the segmentation and pull EM out of poor local maxima. [1]

Assume we have a sequence of characters $C = c_1c_2...c_T$ that we wish to segment into chunks $S = s_1s_2...s_M$, where T is the number of characters in the sequence and M is the number of words in the segmentation which may vary in each iteration with the change of the parameters θ. Here chunks s_i will be chosen from the core lexicon $V_1 = \{s_i, i = 1, ..., |V_1|\}$ or the candidate lexicon $V_2 = \{s_j, j = 1, ..., |V_2|\}$. If we already have the probability distributions $\theta = \{\theta_i | \theta_i = p(s_i), i = 1, ..., |V_1|\}$ defined over the core lexicon and $\phi = \{\phi_j | \phi_j = p(s_j), j = 1, ..., |V_2|\}$ over the candidate lexicon, then we can recover the most likely segmentation of the sequence $C = c_1c_2...c_T$ into chunks $S = s_1s_2...s_M$ as follows. First, for any given segmentation S of C, we can calculate the joint likelihood of S and C by

$$\Pr(S, C|\theta, \phi) = \prod_{i=1}^{M_1} \lambda p(s_i) \prod_{j=1}^{M_2} (1-\lambda) p(s_j) = \lambda^{M_1}(1-\lambda)^{M_2} \prod_{i=1}^{M_1} \theta_i \prod_{j=1}^{M_2} \phi_j$$

where M_1 is the number of chunks occurring in the core lexicon, M_2 is the number of chunks occurring in the candidate lexicon, s_k can come from either lexicon, (Note that each chunk s_k must come from exactly one of the core or candidate lexicons.) and λ is the weight of core lexicon. Note that M_1 and M_2 may vary in each iteration with the change

[1]To increase the influence of core words in determining segmentations and allow them to act as more effective guides in processing the training sequence, we assign λ weight to the influence coming from the core lexicon words. In practice, we found $\lambda = 1/2$ is a good choice. The amount 1/2 is set arbitrarily and it is not guaranteed to be optimal. The optimal value of the weight could be determined automatically. However, this would make the algorithms more complicated.

of parameters θ and ϕ.

Our task is to find the segmentation S^* that achieves the maximum likelihood:

$$\begin{aligned} S^* &= \arg\max_S \Pr(S|C;\theta,\phi) && (5.37) \\ &= \arg\max_S \Pr(S,C|\theta,\phi) && (5.38) \end{aligned}$$

Similar to standard EM segmentation, following [38], the update Q function that we use in the EM update is given by

$$Q(k,k+1) = \sum_S \Pr(S|C;\theta^k,\phi^k)\log(\Pr(C,S|\theta^{k+1},\phi^{k+1})) \quad (5.39)$$

Maximizing (5.39) under the constraints that $\sum_i \theta_i^{k+1} = 1$ and $\sum_j \phi_j^{k+1} = 1$ yields the parameter re-estimation formulas

$$\theta_i^{k+1} = \frac{\sum_S \#(s_i,S) \times \Pr(S,C|\theta^k,\phi^k)}{\sum_{s_i}\sum_S \#(s_i,S) \times \Pr(S,C|\theta^k,\phi^k)} \quad (5.40)$$

$$\phi_j^{k+1} = \frac{\sum_S \#(s_j,S) \times \Pr(S,C|\theta^k,\phi^k)}{\sum_{s_j}\sum_S \#(s_j,S) \times \Pr(S,C|\theta^k,\phi^k)} \quad (5.41)$$

where $\#(s_i,S)$ is the number of times s_i occurs in the segmentation S. These are the standard re-estimation formulas, and are the same for θ and ϕ except that each will be reinitialized differently in successive optimizations (see below).

In both cases the denominator is a weighted sum of the number of words in all possible segmentations, the numerator is a normalization constant, and Equ. (5.40) and Equ. (5.41) therefore are weighted frequency counts. Thus, the updates can be efficiently calculated using the forward and backward algorithm, or efficiently approximated using the Viterbi algorithm.

The main difficulty with applying EM to this problem is that the probability distributions are complex and typically cause EM to get trapped in poor local maxima. To help guide EM to better probability distributions, we partition the lexicon into a core and candidate lexicon, V_1 and V_2; where V_1 is initialized to be empty and V_2 contains all words. In a first pass, starting from the uniform distribution, EM is used to increase the likelihood of the training corpus C_1. When the training process stabilizes, the N words with highest probability are selected from V_2 and moved to V_1, after which all the probabilities are rescaled so that V_1 and V_2 each contain half the total probability mass. EM is then run again. The rationale for shifting half of the probability mass to V_1 is that this increases the influence of core words in determining segmentations and allows them to act as more effective guides in processing the training sequence. We call this procedure of successively moving the top N words to V_1 *forward selection.*

Forward selection is repeated until the segmentation performance of Viterbi on the validation corpus C_2 leads to a decrease in F-measure (which means we must have included some erroneous core words). After forward selection terminates, N is decremented by 5, and we carry out a process of *backward deletion*, where the N words with the lowest probability in V_1 are moved back to V_2, and EM training is successively repeated until F-measure again decreases on the validation corpus C_2 (which means we must have deleted some correct core words). The two procedures of forward selection and backward deletion are alternated, decrementing N by 5 at each alternation, until $N \leq 0$; as shown in Figure 5.5. As with EM, the outcome of this self-supervised training procedure is a probability distribution over the lexicon that can be used by Viterbi to segment test sequences.

Also, after EM training comes stable, we employ a mutual information based lexicon pruning to prune illegal words (see Section 5.4), which is found to be important in segmentation.

5.6.1 Experimental Results

We use a training corpus, C_1, that consists of 90M (5,032,168 sentences) of unsegmented Chinese characters supplied by the author of [49], which contains one year of the "People's Daily" news service stories (www.snweb.com). The test corpus C_3 is ChineseTreebank from LDC[2] (1M, 10,730 sentences), which contains 325 articles from "Xinhua newswire" between 1994 and 1998 that have been correctly segmented by hand. We remove all white-space from C_3 and create an unsegmented corpus C_3''. We then use the algorithm described in Section 5.6 to recover C_3' from C_3''. The validation corpus, C_2, consists of 2000 sentences randomly selected from the test corpus.

According to the *1980 Frequency Dictionary of Modern Chinese* (c.f. [48]), the 9000 most frequent words in Chinese consist of 26.7% uni-grams, 69.8% bi-grams, 2.7% tri-grams, 0.007% 4-grams, and 0.002% 5-grams. So in our model, we limit the length of Chinese words to 4 characters, which is sufficient for most situations.

One sample of unsegmented raw text from the Chinese Treebank (the first article) and the corresponding segmentation output are shown in Figure 5.6 and Figure 5.7 respectively. The experimental results are shown in Table 5.4. The first set of results, Results 1, are obtained by standard EM training, Results 2 are obtained by self-supervised training, Re-

[2]http://www.ldc.upenn.edu/ctb/

```
0. Input
   Unsegmented training corpus $C_1$
   Validation corpus $C_2$

1. Initialize
   $V_1$ = empty;
   $V_2$ contains all potential words;
   $OldFmeasure = -\infty$;
   $bForwardSelection$=true;
   set $N$ to a fixed number;

2. Iterate
   while ($N > 0$){// $N = 50$ in experiments
     EM based on current $V_1$ and $V_2$ until convergence;
     Calculate $NewFmeasure$ on validation corpus $C_2$;
     if($NewFmeasure < OldFmeasure$){
       // change selection direction
       $bForwardSelection = \neg bForwardSelection$;
       $N = N$-5;
     }
     $OldFmeasure = NewFmeasure$;
     //SelectCoreWords(true) performs forward selection
     //SelectCoreWords(false) performs backward selection
     SelectCoreWords($bForwardSelection$);
   }

3. Test
   Test on test corpus $C_3$
```

Figure 5.5: Self-supervised Learning

上海浦东开发与法制建设同步新华社上海二月十日电记者谢金虎张持坚上海浦东近
年来颁布实行了涉及经济贸易建设规划科技文教等领域的七十一件法规性文件确保
了浦东开发的有序进行浦东开发开放是一项振兴上海建设现代化经济贸易金融中心
的跨世纪工程因此大量出现的是以前不曾遇到过的新情况新问题对此浦东不是简单
的采取干一段时间等积累了经验以后再制定法规条例的方式而是借鉴发达国家和广
州等特区的经验教训聘请国内外有关专家学者积极及时地制定和推出法规性文件使
这些经济活动一出现就被纳入法制轨道去年初浦东新区诞生的中国第一家医疗机构
药品采购服务中心正因为一开始就比较规范运转至今成交药品一亿多元没有发现一
例回扣建筑是开发浦东的一项主要经济活动这些年有数百家建筑公司四千余个建筑
工地遍布在这片热土上为规范建筑行为防止出现无序现象新区管委会根据国家和上
海市的有关规定结合浦东开发实际及时出台了一系列规范建设市场的文件其中包括
工程施工招投标管理办法拆迁若干规定整治违章建筑实施办法通信设施及管线配套
建设意见建设工地施工环境管理暂行办法等基本到了每个环节都有明确而又具体的
规定建筑公司进区有关部门先送上这些法规性文件然后有专门队伍进行监督检查尽
管浦东新区制定的法规性文件有些比较粗有些还只是暂行规定有待在实践中逐步完
善但这种法制紧跟经济和社会活动受到了国内外投资者的好评他们认为到浦东新区
投资办事有章法讲规矩利益能得到保障

Figure 5.6: Sample unsegmented raw text from the Chinese Treebank

sults 3 are obtained by repeatedly applying lexicon pruning to standard EM training, and Results 4 are obtained by repeatedly applying lexicon pruning to self-supervised training. The row labeled Perfect lexicon is obtained by using the dictionary based segmentation approach (see Section 5.6.2) with the complete lexicon which contains exactly all the words from the test corpus. Soft-counting is the result of [49],[3] which is also a EM based unsupervised learning algorithm. The word-based results are from [112] which uses a suffix tree model. Finally, the Perfect lexicon [112] results are obtained using a lexicon from another test corpus.

There are several observations to make from these results. First, self-supervised training improves the performance of standard EM training from 40% to 54.1%. Second, mutual

[3] [49] did not supply the F-measure, so we calculate it for comparison.

```
上海| 浦东| 开发| 与| 法制| 建设| 同步| 新华社| 上海| 二|月 十日 电| 记者|
谢金虎| 张持坚| 上海| 浦东| 近|年 来| 颁布| 实行| 了| 涉及| 经济| 贸易| 建
设| 规划| 科技| 文教 等| 领域 的| 七十|一 件| 法|规性 文件| 确保| 了| 浦东|
开发| 的| 有序| 进行| 浦东| 开发| 开放| 是| 一 项| 振兴| 上海| 建设| 现代|化|
经济| 贸易| 金融| 中心| 的| 跨|世纪| 工程| 因此| 大量| 出现| 的| 是| 以前|
不| 曾 遇到 过| 的 新| 情况| 新| 问题| 对 此| 浦东| 不是| 简单| 的| 采取|
干| 一段| 时间| 等| 积累| 了| 经验| 以后| 再| 制定| 法规| 条例| 的| 方式| 而
是| 借鉴| 发达| 国家| 和| 广州| 等| 特区| 的| 经验| 教训| 聘请| 国内|外| 有
关| 专家| 学者| 积极| 及时| 地| 制定| 和| 推出| 法|规性 文件| 使| 这些| 经
济| 活动| 一| 出现| 就被| 纳入| 法制| 轨道| 去年| 初| 浦东| 新区| 诞生| 的|
中国| 第一| 家| 医疗| 机构| 药品| 采购| 服务| 中心| 正| 因为| 一 开始 就|
比较| 规范| 运转| 至今| 成交| 药品| 一|亿| 多 元| 没有| 发现| 一例| 回扣| 建
筑| 是| 开发| 浦东| 的| 一项| 主要| 经济| 活动| 这些| 年| 有 数百 家| 建筑|
公司| 四千|余 个| 建筑| 工地| 遍布| 在 这| 片 热土 上| 为| 规范| 建筑| 行
为| 防止| 出现| 无序| 现象| 新区| 管委|会| 根据| 国家| 和| 上海|市| 的| 有关|
规定| 结合| 浦东| 开发| 实际| 及时| 出台| 了 一| 系列| 规范| 建设| 市场| 的|
文件| 其中| 包括| 工程| 施工| 招投标| 管理| 办法| 拆迁| 若干| 规定| 整治|
违章| 建筑| 实施| 办法| 通信| 设施| 及| 管线| 配套| 建设| 意见| 建设| 工地|
施工| 环境| 管理| 暂行| 办法| 等| 基本| 到 了 每| 个 环节 都| 有| 明确| 而
又| 具体| 的| 规定| 建筑| 公司 进区| 有关| 部门| 先 送|上 这些| 法|规性 文
件| 然后| 有| 专门| 队伍| 进行| 监督| 检查| 尽管| 浦东| 新区| 制定| 的| 法|规
性 文件| 有些| 比较| 粗| 有些| 还| 只是| 暂行| 规定| 有待| 在 实践 中| 逐
步| 完善| 但| 这 种| 法制| 紧跟| 经济| 和| 社会| 活动| 受到| 了| 国内|外| 投
资|者 的| 好 评| 他们| 认为| 到| 浦东| 新区| 投资| 办事| 有| 章法| 讲| 规矩|
利益| 能| 得到| 保障|
```

Figure 5.7: Segmentation output of the sample raw text, where the whitespace indicates the true boundaries and the vertical bars indicate the predicated boundaries.

	p_w	r_w	F_w	final lex size
Results 1	44.6%	37.3%	40.0%	19044012
Results 2	55.7%	53.9%	54.1%	19044012
Results 3	73.2%	71.7%	72.1%	1670317
Results 4	75.1%	74.0%	74.2%	1670317
Perfect lexicon	92.2%	91.8%	91.9%	10363
Soft-counting [49]	71.9%	65.7%	68.7%	
Word-based [112]	84.4%	87.8%	86.0%	
Perfect lexicon [112]	95.9%	93.6%	94.7%	

Table 5.4: Experimental results

information pruning gives even greater improvement (from 40% to 72.1%), verifying the claim of [112] that the lexicon is more influential than the model itself. Lexicon pruning reduces the lexicon size from 19044012 to 1670317, which makes the lexicon much smaller. By combining the two strategies, we obtain further improvement (from 40% to 74.2%), which is promising performance considering that we are using a purely unsupervised approach. By comparison, the result given by a perfect lexicon is 91.9%. Finally, the two improvement strategies of self-supervised training and lexicon pruning are not entirely complementary and therefore the resulting performance does not increase additively when both are combined (72.1% using lexicon pruning alone to 74.2% using both).

A direct comparison can be made to [49] because it also investigates a purely unsupervised training approach without exploiting any additional context information and uses the same training set we have considered. When we compare the results, we find that self-supervised training plus lexicon pruning achieves both a higher precision and recall, and obtains a 5.5% (from 68.7% to 74.2%) improvement in F-measure. One problem with this comparison, however, is that Ge et al. [49] does not report precisely how the testing was performed. It is also possible to compare to [112], which uses a suffix tree model and employs context information (high entropy surroundings) to achieve an 86% F-measure score. This result suggests that context information is very important. However, because of a different test set (our test set is the 1M ChineseTreebank data set from LDC, whereas their test set is 61K of data pre-segmented by the NMSU segmenter [69] and corrected by hand), the comparison is not completely calibrated. In the perfect lexicon experiments, Ponte and Croft [112] achieves higher performance (94.7% F-measure), whereas only 91.9% is achieved in our experiments. This suggests that we may obtain better performance when testing on the data used in [112]. Nevertheless, the result of [112] appears to be quite strong, and demonstrates the utility of using local context rather than assuming independence between words, as we have done.

The results are promising because we have not yet employed any smoothing techniques and local context information which were proved to be very useful [13, 15, 112]. We believe that after adding these techniques, the performance will be dramatically increased.

Also, in the experiments (numbers not reported), we find that higher likelihood does not necessary mean better segmentation. EM and self-supervised segmentation only consider the frequencies of chunks, whereas highly frequent chunks may not be true words. We are currently considering using language characteristics such as part of speech (POS) information to augment segmentation.

5.6.2 Comparison to Supervised Word Segmentation

So far we have mainly focused on unsupervised word segmentation. Our results improve previous results obtained by unsupervised learning methods. However, not surprisingly, unsupervised methods are not as effective as supervised learning methods in this domain. Two accurate word segmentation methods based on supervised learning have been developed for Chinese: the dictionary based approach and the compression based approach [139].

The dictionary based approach is the most popular Chinese word segmentation method. Given a manually built dictionary, a heuristic method, such as longest word match, is then used to segment new sequences. In our experiments we used the longest forward match method in which text is scanned sequentially and the longest matching word from the dictionary is taken at each successive location. Using a dictionary that contains 69,353 words and phrases [6], we can obtain an 85% accuracy. In Table 5.4, we can obtain 91.9% accuracy when using a *perfect* lexicon. However, it is not possible to obtain such a perfect lexicon in practice.

The PPM (prediction by partial matching) word segmentation algorithm of Teahan et al. [139] is based on the text compression method of Cleary and Witten [31]. PPM learns an n-gram language model by supervised training on a given set of hand segmented Chinese text. To segment a new sentence, PPM seeks the segmentation that gives the best compression using the learned model. By controlling the amount of training data and the order of the language model, we can control the resulting word segmentation accuracy of PPM models. It has been shown to be able to yield up to 95% accuracy [139].

The accuracies obtained by the supervised methods are clearly superior to the results obtained by our state of the art unsupervised segmentation algorithms. However, our unsupervised methods for word segmentation obtain many of the advantages indicated in Section 5.1. Moreover, as we will demonstrate in Chapter 6, our self-supervised word segmentation algorithm yields advantages in Chinese text retrieval over supervised methods.

5.6.3 Error Analysis

Figure 5.8 shows two categories of error that are typically committed by our model. In each case, the left string shows the correct word and the right bracketed string shows the recovered word segmentation. The first error category is *date and number.* In Chinese, dates and numbers are represented by 10 characters. Unlike Arab numbers, the 10 Chinese numeric characters are not different from other Chinese characters. Most dates and numbers are not correctly recognized because they do not have sufficient joint statistics in the training corpus to prevent them from being broken down into some smaller numbers. For example, *"1937 year"* is broken into *"19", "3", "7 year"*; *"2 wan 3 qian 1 bai 1 shi 4"* is broken into *"2 wan", "3 qian", "1", "bai 1 shi 4"* (*wan* denotes 10-thousand, *qian* denotes thousand, *bai* denotes hundred, and *shi* denotes ten). It turns out that one can easily use heuristic methods to correct errors in these special cases. For example, if a string of concatenated characters are all numeric characters, then the string is very likely to be a single date or number.

The second error category is the recognition of compound nouns. For example, the compound *"united nation"* is recovered as two words *"union"* and *"country"*; *"team Australia"* is recovered to *"team"* and *"Australia"*. One reason for the failure in correctly recovering compounds is that we are limiting the maximum length of a word to 4 characters, whereas many compounds are longer than this limit. However, simply relaxing the length limit creates significant sparse data problems. The recognition of noun compounds appears to be difficult, and a general approach may have to consider language constraints.

A: Date and Numbers	
一九三七年	(一九 三 七年)
二万三千一百一十四	(二万 三千 一 百一十四)
B: Compounds	
联合国	(联合 国)
澳大利亚队	(澳大利亚 队)

Figure 5.8: Typical Chinese segmentation errors

5.7 Relation to Previous Research

Our work is related to many other research efforts.

Word boundary detection and lexical acquisition in continuous speech: There is some previous research on detecting words from continuous speech using a hierarchical structure. In addition to the work of de Marcken [36] mentioned in Section 5.5.3, Witten and Nevill-Manning [95] also use a hierarchical approach to detect boundaries, but similarly do not directly report word boundaries. Our model is limited to 2 levels and therefore explicitly identifies word boundaries.

The work of [76, 60, 14, 13] is all based on the MDL principle, but these investigations differ in how the description length is encoded. Kit and Wilks [76] use a description length gain measure, but do not use hierarchical structure nor EM to learn a lexicon. [14] and [13] use Huffman codes to describe words and use a generative model in which the size of lexicon is predefined, and then use this lexicon to generate observations. Our model uses a simple lexicon pruning scheme to automatically determine the size of the lexicon.

[14, 13, 28] test their algorithms on phonemically transcribed corpora (the CHILDES collection and spontaneous child-directed British English) but in practice the phonemes are not explicitly identified in the utterances, and therefore the basic unit in speech is the phone. Here it is also necessary to detect phonemes automatically from a phone sequence. In essence, this corresponds to the second level of our model: learning words given phonemes.

Unsupervised Chinese word segmentation: Ge et al. [49] use a soft counting version of EM to learn to segment Chinese. To augment the influence of important words, they shift the probability mass to likely words by soft counting. In our model, we shift half of the probability space to core words by dividing the lexicon to two parts. Also, they do not employ any sort of lexicon pruning, which we have found is essential to improving performance. Ponte and Croft [112] uses a suffix tree word-based model and a bi-gram model to segment Chinese strings. This work takes the surrounding word information into consideration when constructing the lexicon. They uses a more complicated Hidden Markov Model (HMM) model that includes special recognizers for Chinese names and a component for morphologically derived words. As pointed out by Ge et al. [49], standard EM segmentation can be thought of as a zero order HMM.

Morphology learning: In the first stage of our hierarchical model we learn morphemes from character strings. There has been a lot of work on automated morphology learning; see [51] for a recent survey. However, unlike other morphology learning work, the goal of our model is not to learn morphemes per se, but to use morphemes to identify word boundaries. Therefore, the morphemes learned in our first stage may not be true morphemes in the full linguistic sense [54, 89]—they are really just chunks that occur most often in words.

Hierarchical models for sparse data: Many authors have proposed hierarchical structures to reduce the effects of sparse training data. Freitag and McCallum [46] use hierarchical models to regularize hidden Markov model emission probability estimates. Baker et al.

[5] use hierarchical models to regularize naive Bayes models for topic detection and tracking. Slonim and Tishby [130] use a similar two-step method to cluster documents. In their work, Slonim and Tishby first cluster words in a document, and then cluster documents based on the word clusters obtained from the first step. The main technique for exploiting a hierarchy in these cases is to share common information between related clusters of data. As long as there is common information, a hierarchical structure should prove beneficial. In our case, the common information is the morphemes that are shared between words.

Mutual information lexicon optimization: Sproat et al. [134] use mutual information to build a lexicon, but deal with words up to 2 characters only. Ponte and Croft [112] and Zhao et al. [154] use mutual information and context information to build a lexicon based on the statistics directly obtained from a training corpus. Instead of building a lexicon from scratch, we first add all possible words into the lexicon and then use mutual information to prune the illegal words after training by EM. Therefore, the statistics we use for calculating mutual information are more reliable than those directly obtained from corpus by frequency counting. Zhao et al. [154] also use mutual information to optimize a Chinese lexicon and test the effect of some variations of mutual information on Chinese lexicon optimization. Their results show that the basic mutual information formula $\log \frac{Pr(x,y)}{Pr(x) \times Pr(y)}$ does not work well in every circumstance, and some variations are suggested for different applications. In our work, we correct the basic formula by using a coefficient that is related to the length of the word.

5.8 Summary

In this chapter, we present several methods to alleviate the sparse data problem in unsupervised word segmentation.

First, we propose a mutual information based lexicon pruning scheme to greatly reduce the lexicon size and help guide EM out of local training maxima.

Second, we presented a two-level hierarchical EM approach to word segmentation and word discovery by exploiting the internal structure of English words. The hierarchical structure we impose is natural and effectively deals with the sparse training data problem. Our model can learn phonemes and words automatically. We tested our model on the Brown corpus and obtained a noticeable improvement over MDL based methods on the same data. Overall, these results show the potential advantage of hierarchical models for speech segmentation and text-to-speech synthesis, as well as compound and phrase detection in natural language processing.

Third, we describe a new unsupervised method for discovering Chinese words from an unsegmented corpus. Combined with an efficient mutual information based lexicon pruning scheme, we achieved competitive results. It is worthwhile to point out that the self-supervised method does not apply only to Chinese, but also applies to other languages.

There remain many open questions. As shown in many other areas of research, MDL or Bayesian estimators often yield better models than a straightforward maximum likelihood approach. It is therefore worthwhile to consider imposing a hierarchical model on description length gain [76] or exploit a prior in EM training. Also, our use of EM sets the probability of all non-occurring words in the training sequence to zero. It would be beneficial to employ the smoothing techniques we discussed in Chapter 3 in estimating these probabilities, although that will make the algorithm much more complicated. Another problem with our self-supervised training procedure is that it puts equal weight on the core and candidate lexicons. One interesting idea would be to automatically estimate the weights of the two lexicons by using a mixture model.

In our experiments, we found that high likelihood did not always correspond to accurate segmentation performance. The reason is that EM segmentation only gives the most likely segmentation, where the chunks may not be true words in a language, although they have high frequency in the training set. That is, in Equ. (5.29), we did not consider any language characteristics that may be helpful in segmentation. We plan to use part of speech (POS) information to augment word segmentation. Here, we are looking for a segmentation that gives not only the most likely boundaries, but also the mostly likely POS at the same time

$$S^* = \arg\max_S \Pr(S, T|C; \theta) \tag{5.42}$$

where S, T, C, θ are the segmentation, POS tag, character string, and probability distribution defined over the lexicon respectively. If we assume T is independent of C, we have

$$\Pr(S, T|C; \theta) = \Pr(S|C; \theta) \Pr(T|S; \theta) \tag{5.43}$$

The first part $\Pr(S|C;\theta)$ is just the standard segmentation that we have already solved. The second part $\Pr(T|S;\theta)$ is the POS tagger which gives the most likely POS tag sequence given the segmentation S. So, there are two steps in the segmentation. First, we segment the character string C into chunks S, then compute the probability of POS tags T given these chunks S. The search for the best segmentation is more complicated than standard segmentation, since we have to consider two parts: $\Pr(S|C;\theta)$ and $\Pr(T|S;\theta)$.

Chapter 6

Applying Unsupervised Word Segmentation to Chinese Text Retrieval

In this chapter, we discuss how to use self-supervised word segmentation to improve Chinese text retrieval. We also investigate the relationship between retrieval performance and word segmentation performance for Chinese text retrieval.

6.1 Introduction

Increasing interest in cross-lingual and multilingual information retrieval has created the challenge of designing accurate information retrieval systems for Asian languages such as Chinese, Thai and Japanese. For multilingual information retrieval it is important to have an adaptable system which can be easily ported to new domains and languages. However, in designing information retrieval systems for these languages one faces the challenge of addressing the word segmentation problem as part of the retrieval process. This creates significant problems both in interpreting queries and in indexing the text corpus.

In Chinese text retrieval, the first step is to tokenize the collection. Traditionally there have been three approaches to tokenization: the dictionary based approach, the character based approach and the mutual information based approach [24, 64, 99, 100]. The dictionary based approach has already been introduced in Section 5.6.1. In the dictionary based approach, one pre-defines a lexicon containing a large number of Chinese words and then uses heuristic methods such as maximum matching to segment Chinese sentences. In the character based approach, sentences are tokenized simply by taking each character to be a basic unit. In the mutual information based approach, one uses the statistics of Chinese characters in the corpus to mark word boundaries. The lexical statistics include the occurrence frequency of each character in the corpus, and the co-occurrence frequency of each pair of characters in the corpus. All three approaches have advantages and disadvantages.

The dictionary based approach is the oldest method and remains the most widely used one in Chinese information retrieval. It has the advantage of requiring a smaller inverted index file, allowing faster retrieval speed, and also allowing additional linguistic information to be incorporated in the retrieval system (e.g. synonyms). The most prominent disadvantage of the dictionary based approach is that it requires a large pre-defined lexicon, which normally must be constructed by hand with a significant amount of time and labor. Moreover, the lexicon constructed for one language/domain is not portable to another languge/domain, and it is virtually impossible to list all Chinese words in a dictionary since the set of words is open-ended [24], as in any language. An additional shortcoming of the traditional maximum matching method used in the dictionary approach is that a character sequence is always segmented the same way regardless of context, which violates the true nature of Chinese text.

For the character based approach, the most prominent advantage is that it does not require a pre-defined lexicon. Each character is considered as a basic unit. However, the disadvantages include the requirement of a huge index file, much slower retrieval speed, and the fact that it is difficult to incorporate linguistic information of any kind.

Both the dictionary based and the character based approaches have been successfully applied to Chinese information retrieval in recent work [17, 24, 63]. Overall, the character based approach has tended to yield better retrieval precision [18, 64, 148]. Therefore there remains a question of whether word segmentation is necessary at all for Chinese text retrieval. However, some researchers [99, 100] have argued that there exist some inherent difficulties in the character based approach. For example, a modern Chinese information retrieval system should be able to take into account more than just character information, but should also be able to exploit sophisticated techniques such as latent semantic indexing [58]. Thus text segmentation, especially that using machine learning techniques, is still a challenging and interesting problem in Chinese text retrieval [151]. In this vein, Kwok [77] proposed using overlapping bi-grams for word segmentation, but this creates a large index file due to overlapping.

Chen et al [24] proposed using a mutual information based approach for word segmentation and obtained better retrieval performance than overlapping bi-gram segmentation. However, the mutual information based approach limits words to be at most two characters. Although most Chinese words are only one or two characters long, there remain many words that are longer than two characters.

We are interested in determining whether machine learning can be used for Chinese information retrieval instead of manually constructed dictionaries, while achieving comparable performance to the mutual information based approach, the character based approach, and the dictionary based approach to segmentation. In this chapter, we propose using the self-supervised segmentation method introduce in Chapter 5 for Chinese information retrieval. The self-supervised approach has many of the advantages of the other approaches, while overcoming many of their shortcomings. For *self-supervised segmentation*, no predefined lexicon is required. Instead, all that is needed is a large unsegmented training corpus—which is almost always easy to obtain. We automatically learn a lexicon and lexical distribution from the training corpus by using the EM algorithm [38], and then segment the collections using the Viterbi algorithm [114]. Since our segmentation approach is completely unsupervised and language independent, it can be easily adapted to other languages.

We have implemented the mutual information based,[1] character based, dictionary based, and self-supervised methods, and compared their retrieval effectiveness at different word segmentation accuracies. In terms of raw word segmentation accuracy, these methods are not equivalent. For example, the best segmentation accuracy obtained by the self-supervised segmentation is around 77% (on the PH corpus [15, 57, 139] [2]). This is not as good as many supervised learning segmenters [57, 139]. However, in this chapter, we investigate whether this segmentation performance is sufficient for Chinese information

[1]We did not implement the overlapping bi-gram approach because Chen et al. [24] found that the non-overlapping mutual information based approach performed better.

[2]We used the ChineseTreebank as the test data in Chapter 5. We noticed that the performance on PH corpus is slightly higher than that on the ChineseTreebank.

retrieval.

The relationship between Chinese word segmentation accuracy and information retrieval performance has recently been investigated in the literature. Foo and Li [45] have conducted a series of experiments which suggest that the word segmentation approach does indeed have effect on retrieval performance. Specifically, they observe that recognizing words of length two or more can produce better retrieval performance, and the existence of ambiguous words resulting from the word segmentation process can decrease retrieval performance. Similarly, Palmer and Burger [102] observe that accurate segmentation tends to improve retrieval performance. All of this previous research has indicated that there is indeed some sort of correlation between word segmentation accuracy and retrieval performance. However, the nature of this correlation is not well understood.

One reason why the relationship between segmentation and retrieval performance has not been well understood is that previous investigators have not considered using a variety of Chinese word segmenters exhibiting a wide range of segmentation accuracies, from low to high. In order to address this problem, we employ four families of Chinese word segmentation algorithms from the recent literature. The first technique we employed was the standard maximum matching dictionary based approach. The second is the mutual information based approach [24]. The remaining two algorithms were selected because they can both be altered by simple parameter settings to obtain different word segmentation accuracies. Specifically, the third Chinese word segmenter we investigated was the minimum description length algorithm of Teahan et al. [139], and the fourth was the EM based technique introduced in Chapter 5. Overall, these segmenters demonstrate word identification accuracies ranging from 44% to 95% on the PH corpus.

The rest of the chapter is organized as follows. First Section 6.2 briefly describes the information retrieval environment we use. Then we present the experiments we conducted on the TREC-5 and TREC-6 data sets in Sections 6.3, 6.4 and 6.5 respectively. Finally, a discussion and conclusions are given in Section 6.6 and Section 6.7.

6.2 Information Retrieval Environment

We conducted our information retrieval experiments using the OKAPI system [64, 118]. To construct a dictionary based information retrieval system we considered two different single unit weighting functions. They are both extended versions of ICF,[3] which include document length and within-document and within-query frequencies to provide additional evidence. Adding this evidence makes the term-weighting dependent on the document, which has been shown to be highly beneficial in English text retrieval [118].

6.2.1 BM25 Weighting Function

The first function, called BM25 [6], is

$$w = \frac{(k_1+1)*tf}{K+tf} * \log\frac{N-n+0.5}{n+0.5} * \frac{(k_3+1)*qtf}{k_3+qtf} \quad \oplus \quad k_2 * nq * \frac{(avdl-dl)}{(avdl+dl)} \qquad (6.1)$$

where N is the number of indexed documents in the collection, n is the number of documents containing a specific term, *tf* is within-document term frequency, *qtf* is within-query term frequency, *dl* is the length of the document, *avdl* is the average document length,

[3]ICF is defined as $w = \log\frac{N-n+0.5}{n+0.5}$, where N is the number of indexed documents in the collection and n is the number of documents containing a specific term [131].

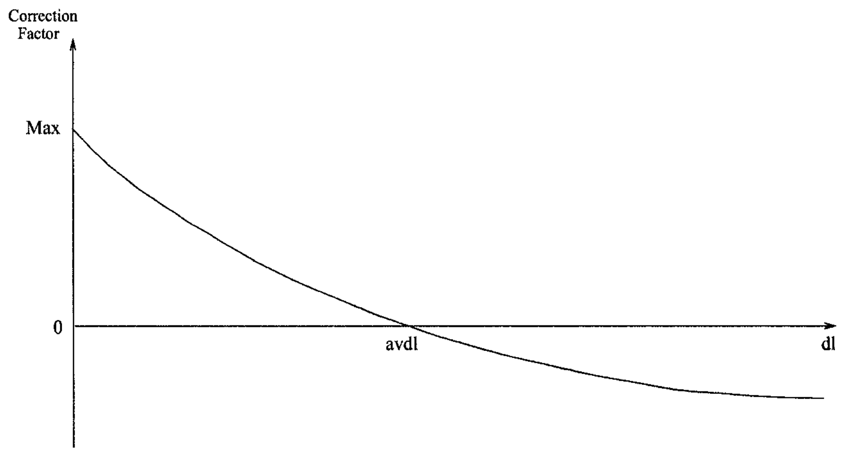

Figure 6.1: Curve for BM25's correction factor

nq is the number of query terms, the k_is are tuning constants (which depend on the database and possibly on the nature of the queries and are empirically determined), K is $k_1 * ((1 - b) + b * dl/avdl)$, and $\oplus$ indicates that its subsequent component is added only once per document, rather than for each term. The component

$$k_2 * nq * \frac{(avdl - dl)}{(avdl + dl)}$$

is called *the correction factor*, which is designed to take into account the length of a document. The value of the correction factor decreases with dl, from a maximum as $dl \rightarrow 0$, at which $dl = avdl$, and to a minimum as $dl \rightarrow \infty$, as shown in Figure 6.1.

This design of the correction factor assumes that the shorter the document, the larger the correction factor should be, i.e., the more likely the document is relevant.

In our experiments, the values of k_1, k_2, k_3 and b in the BM25 function are set to be 2.0, 0, 5.0 and 0.75 respectively. Note that we set k_2 to be 0, which means that the correction factor is not considered. The setting of these numbers was obtained from previous experiments on English text retrieval and from initial experiments on Chinese text retrieval. For example, we found that the system produces better results if we set k_2 to be 0.

6.2.2 BM26 Weighting Function

The fact that improved performance of BM25 is achieved when k_2 is set to be zero in BM25 (i.e., the correction factor is ignored) indicates that the correction factor in BM25 is not designed properly. To repair this problem, we propose an enhanced version of BM25, referred to as BM26, which is based on the following two assumptions: (1) overly short documents are not relevant; (2) the functional curve for the correction factor should be consistent with the distribution of relevant documents in the standard text collection provided by the TREC conferences. More details can be found in [64]. BM26 is defined as follows:

$$w = \frac{(k_1+1)*tf}{K+tf} * \log\frac{N-n+0.5}{n+0.5} * \frac{(k_3+1)*qtf}{k_3+qtf} \quad \oplus \quad k_d * y \tag{6.2}$$

where all the parameters have the same meaning as in BM25 except k_d is a tuning constant and

$$y = \begin{cases} \ln(\frac{dl}{avdl}) + \ln(x_1) & \text{if } 0 < dl <= rel\_avdl; \\ (\ln(\frac{rel\_avdl}{avdl}) + \ln(x_1))(1 - \frac{dl - rel\_avdl}{x_2 * avdl - rel\_avdl}) & \text{if } rel\_avdl < dl < \infty. \end{cases} \tag{6.3}$$

in which dl is the length of the document, $avdl$ is the average document length, rel_avdl is the average relevant document length calculated from previous queries based on the same collection of documents, and x_1 and x_2 are two parameters to be set. The parameter k_d in BM26 is set to have different values in our experiments. When k_d is 0, BM26 becomes

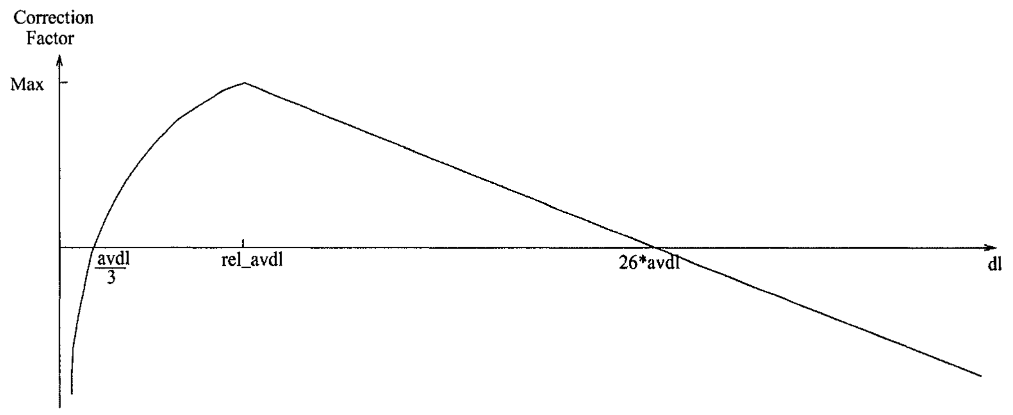

Figure 6.2: Curve for the new correction factor

BM25 since we set the parameter k_2 in BM25 to be 0 in our experiments.

The difference between BM26 and BM25 is the y term of the correction factor. In BM26, y will reach a maximum as $dl \rightarrow rel\_avdl$, and a minimum as $dl \rightarrow 0$ (or $dl \rightarrow \infty$). This relation is shown in Figure 6.2. In our experiments, x_1 and x_2 were set to 3 and 26 respectively.

6.3 Experimental Setup

6.3.1 Data Sets

The test collection we use is from TREC-5 and TREC-6 (the Text REtrieval Conferences) [145]. It contains 164,768 documents and consists of 139,801 articles selected from the *People's Daily* newspaper and 24,988 articles selected from the *Xinhua newswire*, with 0 bytes as the minimum file size, 294,056 bytes as the maximum size and 891 bytes as the average file size. 54 Chinese topics (28 for TREC-5 and 26 for TREC-6) were used in our experiments. The document collection used in TREC-6 Chinese track was identical to the one used in TREC-5. All of the original articles were tagged using SGML. The Chinese characters in these articles were encoded using the GB (GuoBiao) coding scheme.

As mentioned in Section 5.6.1, our self-supervised segmenter is trained on the training set C_1 with validation set C_2, where C_1 is 10M of data which contains a subset of one year of *the People's Daily* news service stories (www.snweb.com), and C_2 is a randomly selected 2000 sentence subset of the Chinese Treebank from LDC, which has been segmented by hand. The parameter k_d in Table 6.1 and Table 6.2 is a tuning constant in BM26 (see equation 6.2). Recall that when k_d is 0, BM26 becomes BM25 in our experiments.

6.3.2 Measuring Retrieval Performance

In our experiments, the TREC relevance judgments for each topic came from the human assessors of the National Institute of Standards and Technology (NIST). Statistical evaluation was done by means of the TREC evaluation program. Several measures are used to evaluate the retrieval result which is an ordered set of retrieved documents. The measures include Average Precision (average precision over 11 recall points, 0.0, 0.1, 0.2,..., 1.0, R Precision (precision after the number of documents retrieved is equal to the number of known relevant documents for a query), and Precision at 100 docs (precision after 100 documents have been retrieved). Detailed descriptions of these measures can be found in [145].

k_d	$L = 2$	$L = 3$	$L = 4$
0	0.3264/0.3639	0.3422/0.3857	0.3246/0.3550
2	0.3326/0.3707	0.3504/0.3901	0.3319/0.3585
6	0.3430/0.3819	0.3613/0.3981	0.3416/0.3692
8	0.3453/0.3849	0.3641/0.3996	0.3419/0.3692
10	0.3450/0.3832	0.3661/0.4027	0.3422/0.3733
15	0.3403/0.3773	0.3601/0.3923	0.3412/0.3747
20	0.3320/0.3744	0.3536/0.3836	0.3368/0.3737
50	0.2756/0.3271	0.2982/0.3444	0.2865/0.3316

Table 6.1: Influence of L on TREC-5 using different weighting methods

6.4 Effects of Self-supervised Segmentation on Chinese Text Retrieval

In this section, we first investigate the influence of maximum word length L on retrieval performance, and then use the optimal length $L = 3$ for further comparison with other term extraction methods.

6.4.1 Influence of Maximum Word Length L

As mentioned in Section 5.6.1, among the 9000 most frequent words in Chinese: 26.7% are uni-grams, 69.8% are bi-grams, 2.7% are tri-grams, 0.007% are 4-grams, and 0.002% are 5-grams. Thus most Chinese words are no longer than 4 characters. In our training algorithm, we set a maximum word length constraint L. To evaluate the effect of L, we experimented with L set to 2, 3 or 4. Table 6.1 and Table 6.2 shows the *average precision/R-precision* results on the TREC-5 and TREC-6 data sets.

Here one can see that the best results were achieved when $L = 3$. An explanation of

k_d	$L = 2$	$L = 3$	$L = 4$
0	0.4363/0.4572	0.4660/0.4849	0.4531/0.4652
2	0.4459/0.4579	0.4754/0.4897	0.4632/0.4718
6	0.4595/0.4693	0.4906/0.4949	0.4781/0.4792
8	0.4635/0.4687	0.4950/0.4939	0.4822/0.4822
10	0.4661/0.4667	0.4968/0.4973	0.4841/0.4839
15	0.4659/0.4685	0.4970/0.5001	0.4820/0.4799
20	0.4603/0.4624	0.4928/0.4976	0.4758/0.4798
50	0.3990/0.4244	0.4186/0.4566	0.4101/0.4459

Table 6.2: Influence of L on TREC-6 using different weighting methods

this observation is that although more than 96% of Chinese words are at most 2 characters long (which is the reason why bi-gram indexing works reasonably well [24, 77]), there are still many words that are longer than 2 characters, and ignoring them compromises the retrieval performance. Thus words longer than 2 characters can still help improve retrieval. This is consistent with [78] where Kwok improves bi-gram indexing by a dictionary of commonly occurring words whose length could be 3 or longer. Obviously, tri-grams and 4-grams will have less ambiguity than bi-grams. However, 4-grams will include many more combinations than tri-grams, which reduces the reliability with which their occurrence probabilities can be estimated. Therefore, statistical over-fitting may explain why $L = 4$ yields worse performance than $L = 3$.

One can also see that the different weighting methods have a large effect on performance. Using the BM26 weighting function causes a large improvement in retrieval quality compared to using BM25 (i.e., $k_d = 0$). Here we find that the performance on the TREC-6 data is much better than on the TREC-5 data.

6.4.2 Comparison with Other Term Extraction Methods

We compare the retrieval performance facilitated by our self-supervised segmentation approach with other traditional segmentation algorithms used in Chinese information retrieval. The first extraction method is the dictionary based method which uses a hand built dictionary of words, compound words, and phrases to index the texts [64]. The second extraction method we compare to is a standard character based approach, in which documents are indexed by single Chinese characters appearing in the text. (However, we would like to emphasize that using single characters for indexing does not imply that we use single characters as keywords for search. For the character based approach, search can be conducted for any multi-character word or phrase identified at search time, whether or not this word or phrase appeared in the dictionary. Therefore the experimental results we report for the character based approach use the character based method fcr indexing and a dictionary based method for topic processing.) The third extraction method we compare to is the mutual information based approach [24], in which one first collects occurrence frequencies for uni-grams and bi-grams, and then uses a mutual information criterion to segment the text [133], and finally uses these segmented terms to index the documents. Finally we also compare to the PPM based method, which has also been briefly discussed in Section 5.6.2

The topic processing method we used in the experiments is simple and automatic. First we rank the words extracted from each topic by the values of their weights multiplied by the within-query frequencies. We then use the 19 top ranked words as retrieval keywords. [4] A detailed description can be found in [64]. The segmentation method used in topic

[4]We chose 19 because it gave the best result among the three numbers we tried in our experiments. There may be a better way to do it. But it does not affect our results.

processing is consistent with that used in document processing except for the character based approach. That is, when the EM method, dictionary based method or mutual information based method are used for document processing, topic processing also uses the EM method, dictionary based method or mutual information based method respectively. However, the *character based method* we use is mixture of a pure character based method and dictionary based method, i.e., it uses the character based method for indexing but uses a dictionary based method for topic processing. This hybrid system yields far superior results to the pure character based method.

The motivation for using the self-supervised segmentation method for Chinese text retrieval is to incorporate the advantages of the character based, dictionary based, and mutual information based approaches, while overcoming their shortcomings. Below we will show these detailed comparisons.

The size of the character based index built for the Chinese TREC collection is about one gigabyte, which is about twice the size of the raw document collection. This is because the positional information about each character's occurrence is stored. In this paper, we use the positional information in the process of retrieval for the character-based approach. However, the size of the character-based index without positional information is much less according to some other experiments [98]. The index for the EM based methods are roughly the same size as for the dictionary based method. The sizes of the index files for the dictionary based, EM based, mutual information (MI) based and PPM based methods are given in Table 6.3. In terms of retrieval time, the EM based methods are similar to the dictionary based methods. The mutual information based method is a little slower than the dictionary based, EM based, and PPM based methods. Each of these four methods is

	index	invert
character	139,734	1,077,393,536
dictionary	1,691,575	678,257,616
EM2	2,087,509	648,472,816
EM3	1,431,104	667,634,000
EM4	1,868,539	677,900,752
MI	11,723,406	730,013,936
PPM90	8,282,064	669,070,144

Table 6.3: Size of Index Files (unit is byte)

about three times faster than the character based approach.

We show the experimental results of the five methods on TREC-5 data in Tables 6.4 and 6.5, and the results on TREC-6 data sets in Tables 6.6 and 6.7. Here the *relevant retrieved* is the number of relevant documents retrieved out of the 2182 or 2958 documents in the collection for TREC-5 or TREC-6 respectively. We set the dictionary based method as the baseline.

On TREC-5 data, we find the EM based segmentation gives a 5.57% improvement in average accuracy over the dictionary based method, but it does a little worse than the character based method. In terms of R-precision, the EM based method yields better performance than all other methods. On TREC-6 data, the EM based method yields slightly worse results than both the dictionary based and the character based methods. On both TREC-5 and TREC-6 datasets, the EM based method produces better results than mutual information based method.

Previous research [17, 64] has suggested that exact segmentation may not be necessary for information retrieval. Our results also support this point: although the segmentation accuracy of the EM based method is not as high as other methods, it yields comparable

Recall	character	dictionary	EM-based	MI-based	PPM-based
0.00	0.7764	0.7681	0.7358	0.7473	0.6655
0.10	0.6243	0.6261	0.6249	0.6263	0.5304
0.20	0.5507	0.5075	0.5429	0.5528	0.4735
0.30	0.4987	0.4531	0.4998	0.4976	0.4309
0.40	0.4458	0.4034	0.4406	0.4289	0.3998
0.50	0.4247	0.3558	0.3954	0.3935	0.3592
0.60	0.3591	0.3180	0.3343	0.3415	0.2972
0.70	0.2711	0.2463	0.2803	0.2569	0.2433
0.80	0.2236	0.1760	0.1859	0.1818	0.1594
0.90	0.1408	0.1154	0.1082	0.0993	0.0914
1.00	0.0266	0.0082	0.0157	0.0275	0.0078
average precision	0.3795	0.3468	0.3661	0.3627	0.3213
improvement	9.43%	baseline	5.57%	4.58%	-7.35%
relevant retrieved	1986	1883	1939	1893	1894

Table 6.4: TREC-5: comparing precision at specified recall rate

retrieval performance to a hand built dictionary approach.

6.5 Relationship between Segmentation Accuracy and Retrieval Performance

The advantages of using EM for word segmentation has in fact been considered in previous research [49, 107]. However, due to the low segmentation accuracies these methods obtain,[5] they still do not tend to be regarded as good methods for Chinese information retrieval. Nevertheless, the results presented so far suggest that this need not be the case. In fact,

[5]The segmentation accuracies of EM based and mutual information based methods, whose retrieval performances shown in Tables 6.4, 6.5, 6.6 and 6.7, are 77% and 61% respectively.

R	character	dictionary	EM-based	MI-based	PPM-based
5	0.5571	0.5429	0.5500	0.5286	0.4571
10	0.5429	0.5143	0.5107	0.5214	0.4821
15	0.4881	0.4810	0.4881	0.4952	0.4643
20	0.4732	0.4732	0.4857	0.4946	0.4500
30	0.4369	0.4321	0.4595	0.4571	0.4238
100	0.3189	0.3150	0.3243	0.3139	0.3007
200	0.2380	0.2302	0.2418	0.2329	0.2291
500	0.1272	0.1216	0.1269	0.1208	0.1202
1000	0.0709	0.0672	0.0693	0.0676	0.0676
R-Precision	0.3963	0.3863	0.4027	0.3988	0.3663
improvement	2.59%	baseline	4.25%	3.24%	-5.18%

Table 6.5: TREC-5: comparing R-precision

we have shown that unsupervised Chinese word segmentation, limited as it is in terms of accuracy, can still facilitate retrieval performance that it comparable with the best current Chinese information retrieval systems. In the following, we investigate the relationship between word segmentation accuracy and retrieval performance in Chinese information retrieval.

6.5.1 Segmentation Accuracy Control

To achieve a wide range of word segmentation accuracies, we employed the various segmenters and varied their parameters to vary the segmentation accuracy. In particular, we used the dictionary based method (see Section 5.6.2), the mutual information based method [24], the PPM based method [139] (also see Section 5.6.2), and the self-supervised method (see Section 5.6).

In the dictionary based approach, we control accuracy by using two different dictio-

Recall	character	dictionary	EM-based	MI-based	PPM-based
0.00	0.9604	0.9558	0.9110	0.9120	0.9149
0.10	0.8144	0.8217	0.8021	0.8105	0.7994
0.20	0.7396	0.7351	0.7482	0.7371	0.7255
0.30	0.6957	0.6586	0.6580	0.6564	0.6486
0.40	0.6662	0.5958	0.5935	0.5921	0.5827
0.50	0.5937	0.5507	0.5267	0.5402	0.5258
0.60	0.5195	0.4708	0.4608	0.4659	0.4692
0.70	0.4284	0.3844	0.3779	0.3683	0.3864
0.80	0.3224	0.2892	0.2738	0.2755	0.2961
0.90	0.1966	0.1485	0.1846	0.1751	0.1796
1.00	0.0239	0.0023	0.0297	0.0050	0.0141
average precision	0.5348	0.5044	0.4970	0.4966	0.4983
improvement	6.03%	baseline	-1.47%	-1.55%	-1.21%
relevant retrieved	2569	2536	2540	2532	2518

Table 6.6: TREC-6: comparing precision at specified recall rate

naries. The first is the Chinese dictionary used by Gey et al. [50], which includes 137,659 entries. The second is the Chinese dictionary used by Beaulieu et al. [6], which contains 69,353 words and phrases. By using the forward maximum matching segmentation strategy with the two dictionaries, Berkeley and City [24, 64], we obtain the segmentation accuracies of 71% and 85% respectively. The segmentation accuracy for the mutual information based approach is 61%.

For the PPM algorithm, by controlling the order of the n-gram language model used (specifically, 2 and 3) we obtain segmenters that achieve 90% and 95% word recognition accuracy respectively.

Finally, for the self-supervised learning technique, by controlling the number of EM iterations and altering the lexicon pruning strategy we obtain word segmentation accuracies

R	character	dictionary	EM-based	MI-based	PPM-based
5	0.7615	0.8077	0.7538	0.7846	0.7846
10	0.7731	0.7885	0.7846	0.7808	0.7615
15	0.7615	0.7667	0.7564	0.7641	0.7410
20	0.7385	0.7404	0.7346	0.7423	0.7096
30	0.6910	0.6936	0.6833	0.6949	0.6769
100	0.5035	0.4923	0.4831	0.4758	0.4838
200	0.3615	0.3521	0.3406	0.3354	0.3456
500	0.1832	0.1808	0.1798	0.1765	0.1782
1000	0.0988	0.0975	0.0977	0.0974	0.0968
R-Precision	0.5404	0.5055	0.5001	0.4955	0.5008
improvement	6.90%	baseline	-1.07%	-1.99%	-0.93%

Table 6.7: TREC-6: comparing R-precision

of 44%, 49%, 53%, 56%, 59%, 61%, 70%, 75%, and 77%.

Thus, overall we obtain 13 different segmenters that achieve segmentation accuracies of 44%, 49%, 53%, 56%, 59%, 61%, 70%, 71%, 75%, 77%, 85%, 90%, and 95%.

6.5.2 Experimental results

Now, given the 13 different segmenters, we conducted extensive experiments on the TREC data sets using different information retrieval methods (achieved by tuning the k_d constant in the term weighting function described in Section 6.2).

Table 6.8 shows the *average precision* and *R-precision* results obtained on the TREC-5 and TREC-6 queries when basing retrieval on word segmentations at 12 different accuracies, for a single retrieval method, $k_d = 10$. To illustrate the results graphically, we re-plot this data in Figure 6.3, in which the x-axis is the segmentation performance and the y-axis is the retrieval performance.

seg. accuracy	TREC-5	TREC-6
44%(EM)	0.2231/0.2843	0.3424/0.3930
49%(EM)	0.2647/0.3259	0.3848/0.4201
53%(EM)	0.2999/0.3376	0.4492/0.4801
56%(EM)	0.3056/0.3462	0.4473/0.4727
59%(EM)	0.3097/0.3533	0.4740/0.4960
61%(MI)	0.3627/0.3988	0.4953/0.4942
70%(EM)	0.3721/0.3988	0.5044/0.5072
71%(Berkeley)	0.3656/0.4088	0.5133/0.5116
75%(EM)	0.3652/0.4000	0.4987/0.5097
77%(EM)	0.3661/0.4027	0.4968/0.4973
85%(City)	0.3468/0.3863	0.5044/0.5055
90%(PPM)	0.3213/0.3663	0.4983/0.5008
95%(PPM)	0.3189/0.3669	0.4867/0.4933

Table 6.8: Average precision and R-precision results on TREC queries when $k_d = 10$.

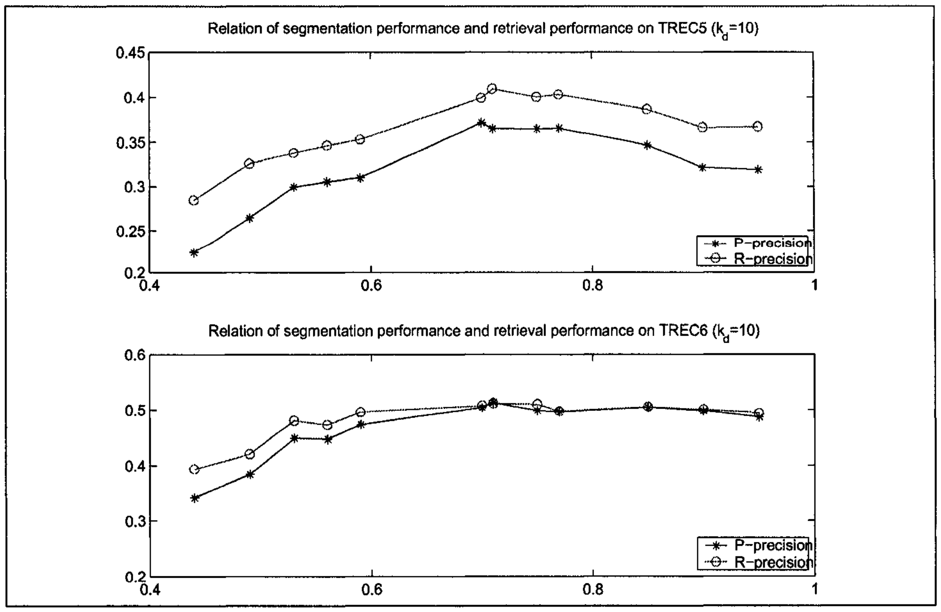

Figure 6.3: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d = 10$.

Clearly these curves demonstrate a nonmonotonic relationship between retrieval performance and segmentation accuracy (on both P-precision and R-precision). In fact, the curves show a clear uni-modal shape, where for segmentation accuracies between 44% and 70% retrieval performance increases steadily, but then plateaus for segmentation accuracies between 70% and 77%, and finally decreases slightly when the segmentation accuracy is increased to 85%, 90% and 95%. This phenomenon is robust to altering the retrieval method by setting $k_d = 0, 6, 8, 15, 20, 50$, as shown in Figures 6.4 to 6.9 respectively.

The highest segmentation accuracy achieved by the self-supervised segmentation method is 77%. However, the best retrieval performance for this method is obtained by setting the segmentation accuracy to 70%.

To give a more detailed picture of the results, Figures 6.10 and 6.11 illustrate the full *precision-recall* curves for $k_d = 10$ at each of the 12 segmentation accuracies, for TREC-5 and TREC-6 queries respectively. In these figures, the results of the 44%, 49% segmentation accuracies are marked with stars, the 53%, 56%, 59% segmentation accuracies are marked with circles, the 70%, 71%, 75%, 77% segmentation accuracies are marked with diamonds, the 85% segmentation accuracy is marked with hexagrams, and the 90% and 95% segmentation accuracies are marked with triangles. We can see that the curves with the diamonds lie above the others, while the curves with stars lie at the lowest positions.

6.6 Related Work and Discussion

In this section, we analyze the experimental results reported in Sections 6.4 and 6.5.

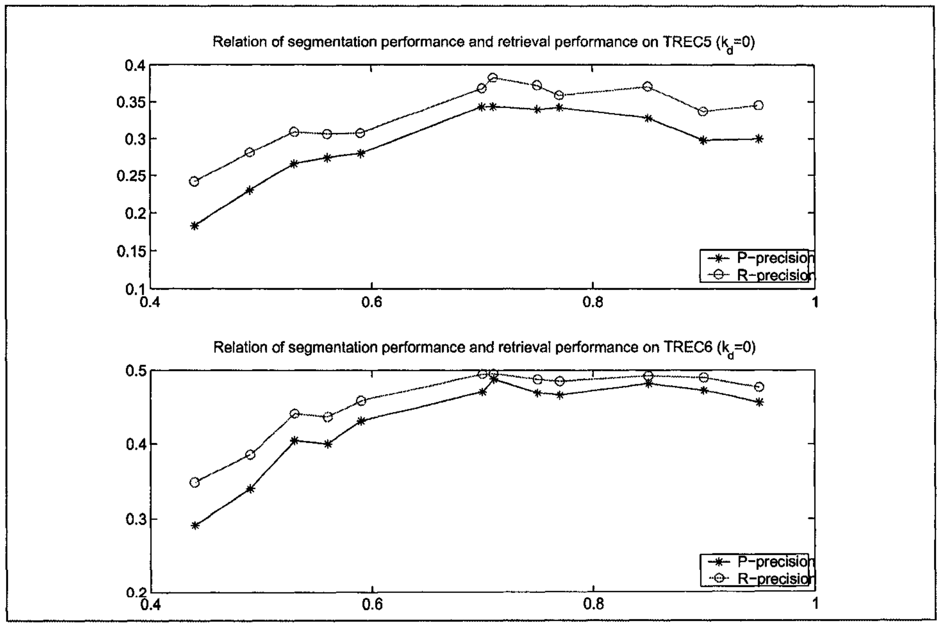

Figure 6.4: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d = 0$.

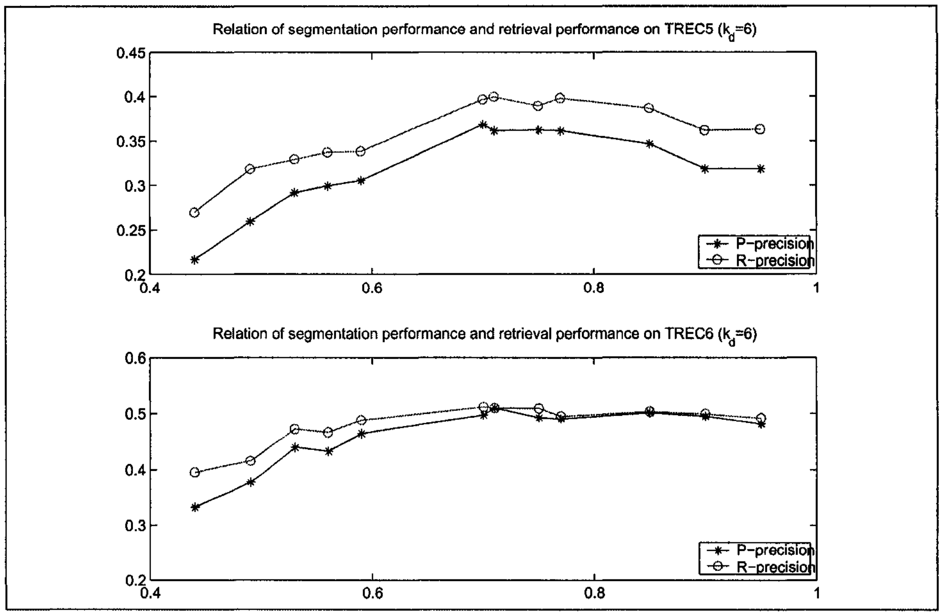

Figure 6.5: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d = 6$.

6.6.1 Word Segmentation in Chinese information retrieval

Our work in this chapter is most closely related to the work of Chen et al. [24]. There it was also proposed that Chinese information retrieval could be conducted without using a manually built dictionary. In their method, Chen et al. collect occurrence frequencies from the corpus, limit the word length to be at most 2 characters, and use the mutual information technique to segment Chinese text. Similarly, we also use frequencies from the corpus and use mutual information during the process. However, our approach differs from theirs in many respects. First we do not limit the word length to 2 characters. The maximum word length could be set arbitrarily to suit the application. In fact, our best results are achieved with $L = 3$. Second, the statistics we used were optimized by an iterative EM process, which is guaranteed to achieve at least a local optimum. This approach should be more

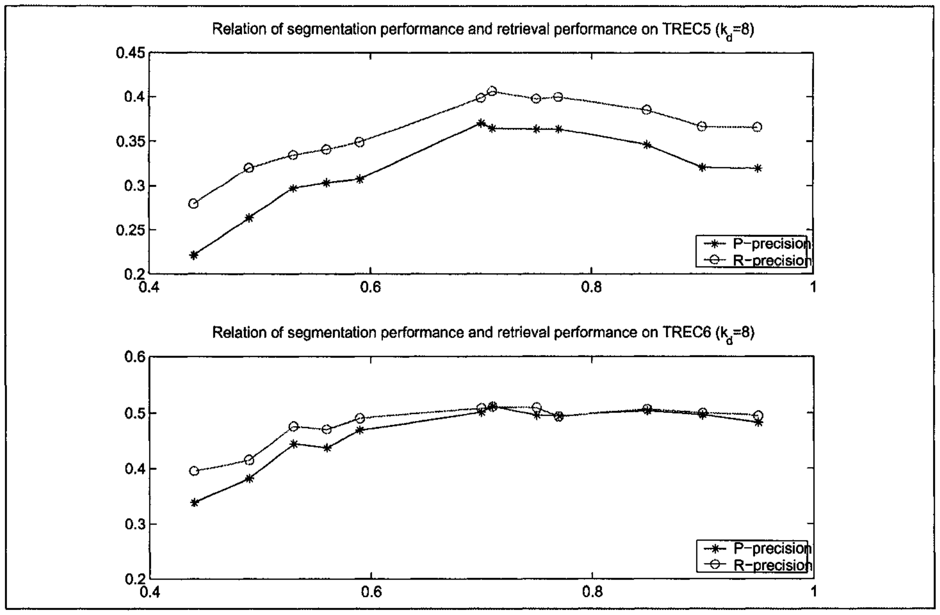

Figure 6.6: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d = 8$.

reliable than using the statistics obtained directly from the corpus. This is confirmed by the experimental results shown in Tables 6.4, 6.5, 6.6 and 6.7. Our work is also strongly related to Kwok [77, 78] who proposed a mixture of uni-grams and bi-grams, or mixture of uni-grams and short words to represent text. The mixture was built with some predefined rules. In our work, this mixture is defined implicitly because our EM based algorithm can segment text into words ranging from 1 character to L characters.

There are three concerns with the EM based approach. The first is that unsupervised learning normally requires a large amount of raw training data to achieve reasonable performance. This raises the question of how much raw data should be enough for training. In our experiments, we used 10M raw data, a subset of 90M data downloaded from *People's Daily* news service stories (www.snweb.com). The entire 90M data set was previously

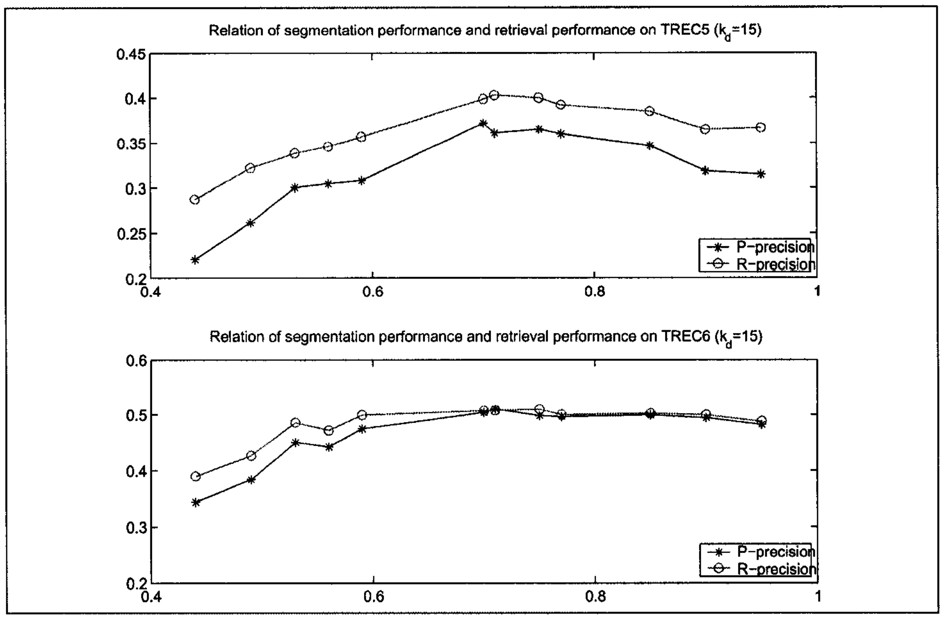

Figure 6.7: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d = 15$.

investigated in [49, 107]. In our experiments, we found that 10M data was sufficient to achieve good performance.

The second concern is that EM training can often take a long time to converge. In our experimental environment (a 750M HZ Pentium IV PC with 2G RAM), it took about 1 hour for a single EM iteration. Ten iterations were sufficient for convergence. After initial training reaches convergence, the lexicon pruning and core lexicon construction routines were then applied. Once lexicon pruning has been carried out, time was reduced to about 30 minutes for a single EM iteration. Overall, we conducted three iterations of lexicon pruning and core lexicon growing. This meant that the entire training process was completed within 36 hours. Note that the training process only has to be conducted once as an off-line preprocessing procedure. Once the lexicon is constructed, the Viterbi

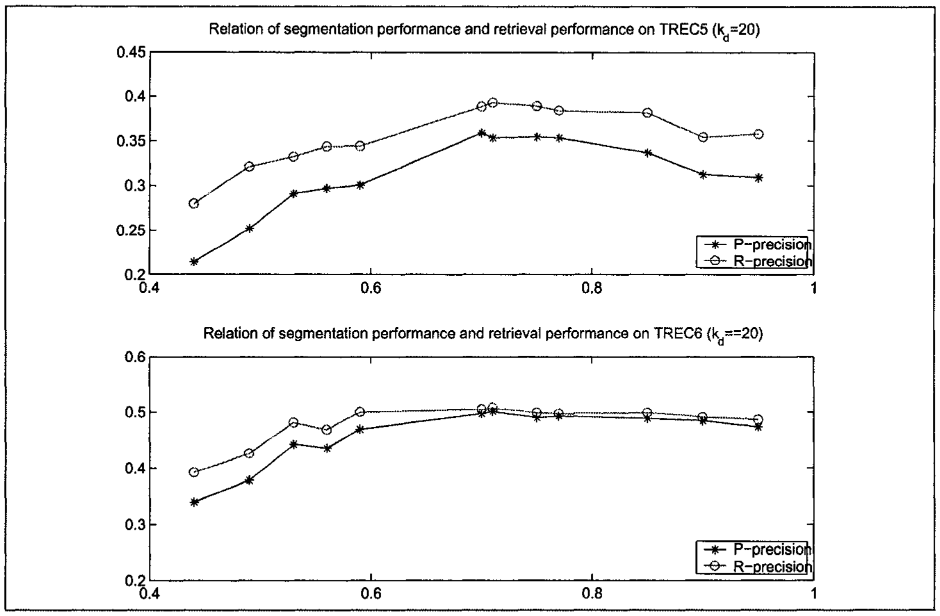

Figure 6.8: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d = 20$.

algorithm can be used to segment each new sentence in time linear in the length of the sentence [114]. Thus, in terms of training time, the unsupervised lexicon construction and word segmentation approach is plausible.

The third concern is that the segmentations obtained by pure unsupervised methods are not as accurate as those obtained by other supervised methods. Such a reduction in segmentation accuracy might not be acceptable in some applications; for example, in machine translation. However, in Chinese text retrieval, we have shown that the segmentation accuracy obtained by unsupervised methods can lead to competitive retrieval performance. It is also worth pointing out that text retrieval in general has to rely on large-scale automatic methods, simply because of the quantity of textual material available. It would be interesting to investigate how much an unsupervised method can help in this matter.

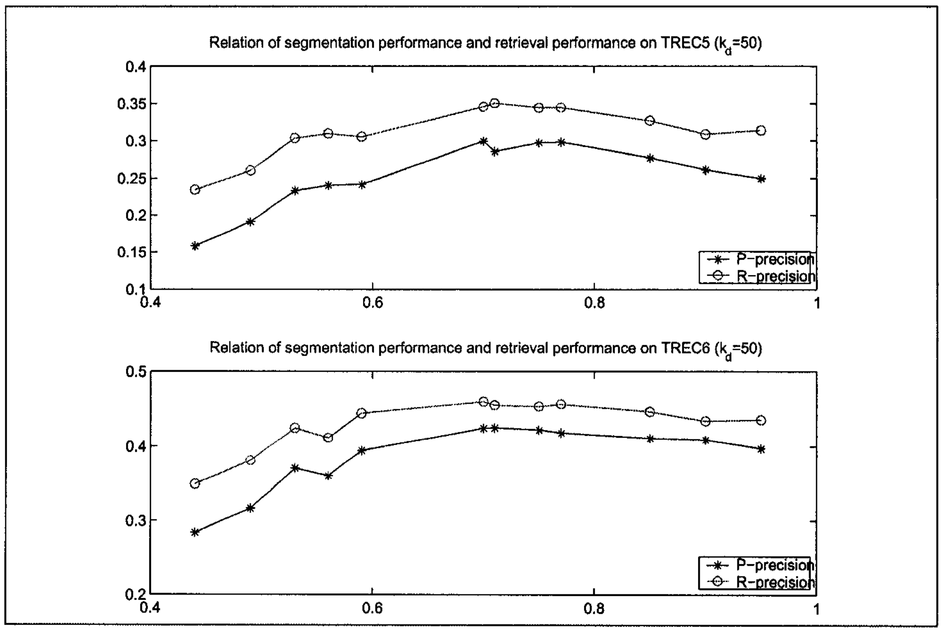

Figure 6.9: Retrieval performance (y-axis) versus segmentation accuracy (x-axis) for $k_d =$ 50.

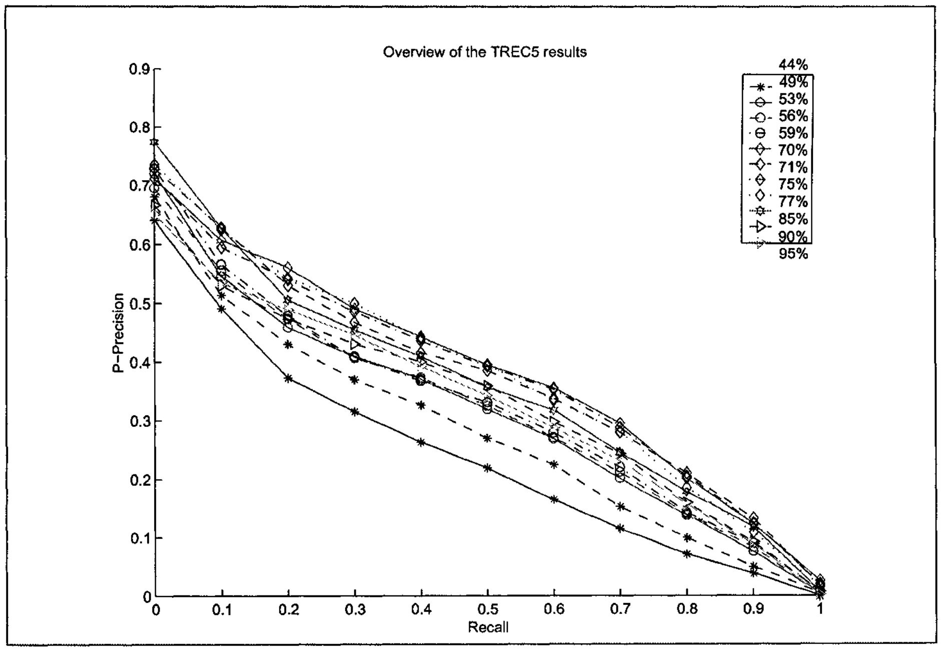

Figure 6.10: TREC5 precision-recall comprehensive view at $k_d = 10$

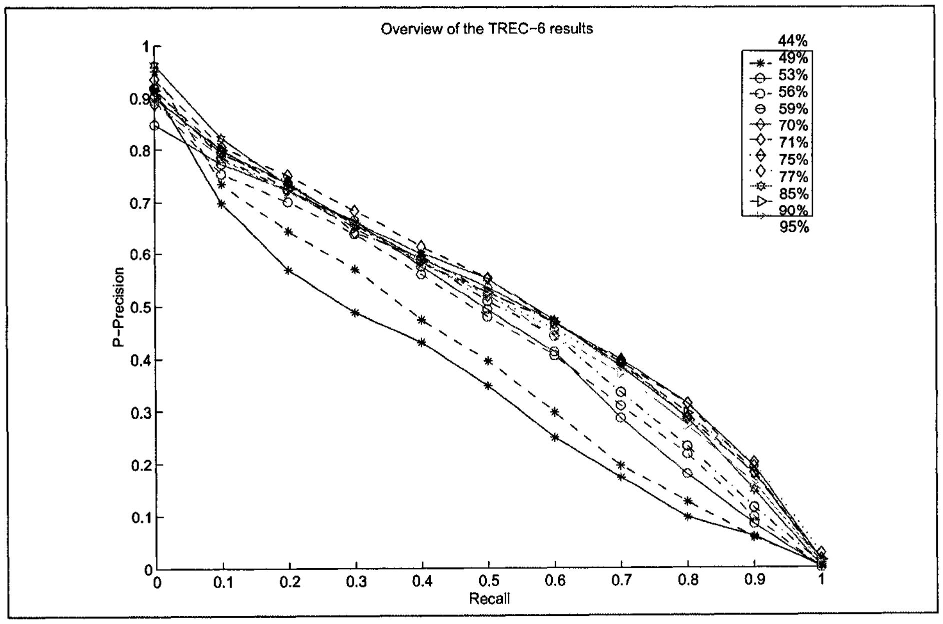

Figure 6.11: TREC6 precision-recall comprehensive view at $k_d = 10$

6.6.2 Relationship between Segmentation Accuracy and Retrieval Performance

The relationship between retrieval performance and segmentation accuracy was surprising to us at first and suggested that there was an interesting phenomenon at work. To attempt to identify the underlying cause of the nonmonotonic relationship, we break the explanation into two parts: an initial phase where retrieval performance increases with increasing segmentation accuracy, and a second effect in the region where retrieval performance plateaus and eventually decreases with increasing segmentation accuracy.

The first part of these performance curves seems easy to explain. At low segmentation accuracies the segmented tokens do not correspond to meaningful linguistic terms, such as words, which hampers retrieval performance (because the term weighting procedure is comparing arbitrary tokens to the query). However, as segmentation accuracy improves, the tokens behave more like true words and the retrieval engine begins to behave more effectively.

However, after a point, when a second regime is reached, retrieval performance no longer increases with improved segmentation accuracy, and eventually begins to decrease. One possible explanation for this is that a weak word segmenter accidentally *breaks* compound words into smaller constituents, and this, surprisingly can yield a benefit for Chinese information retrieval.

For example, one of the test queries, Topic 34, is about the impact of droughts in various regions of China. Retrieval based on the EM-70% segmenter retrieved 84 of the 95 relevant documents in the collection, whereas retrieval based on the PPM-95% segmenter retrieved only 52 relevant documents. In fact, only 2 relevant documents were missed by EM-70%

but retrieved by PPM-95%, whereas 34 documents retrieved by EM-70% and were not retrieved by PPM-95%. In this case, we find that the performance drop appears to be due to the inherent nature of written Chinese. That is, in written Chinese many words can be legally represented by their subparts. For example, 农作物 (agriculture plants) is sometimes represented as 作物 (plants). So for example in Topic 34, the PPM-95% segmenter correctly segments 旱灾 as 旱灾 (drought disaster) and 农作物 correctly as 农作物 (agriculture plants), whereas the EM-70% segmenter incorrectly segments 旱灾 as 旱 (drought) and 灾 (disaster), and incorrectly segments 农作物 as 农 (agriculture) and 作物 (plants). However, by inspecting the relevant documents for Topic 34, we find that there are many Chinese character strings in these documents that are closely related to the correctly segmented word 旱灾 (drought disaster). These alternative words are 春旱，旱魔，受旱，干旱，抗旱，旱区 etc. For example, in the relevant document "pd9105-832", which is ranked 60th by EM-70% and 823rd by PPM-95%, the correctly segmented word 旱灾 does not appear at all. Consequently, the correct segmentation for 旱灾 by PPM-95% leads to a much weaker match than the incorrect segmentation of EM-70%. Here EM-70% segments 旱灾 into 旱 and 灾 , which is not regarded as a correct segmentation. However, it turns out that there are many matches between the topic and relevant documents which contain only 旱. This same phenomenon occurs with the query word 农作物 since many documents only contain the fragment 作物 instead of 农作物, and these documents are all missed by PPM-95% but captured by EM-70%. This explanation is consistent with previous research [79, 80].

The relationship between word segmentation accuracy and retrieval performance is not obvious. The fundamental problem is the balance between specificity and exhaustiveness, or precision and recall. Longer words contribute to increasing precision, but keeping only long words will hurt recall. In Nie et al. [98], to increase precision without penalizing recall, long words are combined with short words and characters contained within long words. However, the traditional dictionary based approach using longest word match for segmentation tends to create long terms, thus it hurts recall.

Chinese word segmentation and text retrieval are two different tasks. In word segmentation research, accuracy is measured by how good a machine segmented string matches a manually segmented string. However, manual segmentation is performed by considering syntactic and semantic information. Whereas for text retrieval, word segmentation is measured by its indirect effect on precision and recall. Since current retrieval methods only consider documents as a bag of words and ignore the syntactic and semantic content of the words, an accurate segmentation may not provide an optimal basis for text retrieval. For example, we observe that the PPM segmenter trained on manually segmented data decreases performance, although it achieves a higher segmentation accuracy. This illustrates the mismatch between word segmentation itself and its use as a tokenizer for text retrieval. The non-monotonic relationship between word segmentation and retrieval performance is caused by this mismatch. This mismatch suggests that different segmentation criteria should be applied in different applications. Our research may provide some guidance in this direction for Chinese text retrieval.

6.7 Summary

We have applied the novel EM based text segmentation method introduced in Chapter 5 for the purposes of Chinese information retrieval, and presented experimental results on recent

TREC data. Our method retains the advantages of the character based, dictionary based and mutual information based methods, while overcoming many of their shortcomings. Although our EM based segmentation method does not yield completely accurate segmentations by itself, it nevertheless performs well as a basis for Chinese information retrieval. We achieve retrieval performance that is comparable (and sometimes even better) than the manual dictionary based, character based and mutual information based segmentation methods. Our results demonstrate that machine learning techniques can be successfully applied to text segmentation and information retrieval to build adaptable systems.

We also observe that the relationship between word segmentation accuracy and retrieval performance is non-monotonic. Retrieval performance first increases as word segmentation accuracy increases, but it begins to plateau after some point and eventually decrease when segmentation accuracy is too high. Our self-supervised word segmentation method yields comparable retrieval performance, although it produces its best result on TREC-5 data by setting the parameters that lead to a segmentation accuracy of only 70%. These observations may also explain the effect of maximum length L (in Section 6.4.1) where we found that $L = 3$ is better than $L = 4$. We conclude that it would be better to bias a segmenter in the direction of deliberately matching shorter rather than longer words. One way to think about this would be to penalize long matches more stringently than short matches in the evaluation.

Although straightforward, these observations suggest a different approach to research on Chinese information retrieval. Instead of focusing on accurate word segmentation, one should pay more attention to issues such as keyword weighting [64] and query keyword extraction [26]. Our current keyword extraction method is very rough, and we are investigating more sophisticated extraction methods such as those used in [24, 26]. Also, since weak unsupervised segmentation can yield better performance in Chinese information retrieval than the other approaches, it seems that machine learning techniques offer a promising new avenue to apply machine learning techniques to information retrieval [132]. Of course, despite these results, we expect that accurate word segmenters will still play an important role in other Chinese information processing tasks, such as information extraction and machine translation. Our research suggests that different criteria for Chinese word segmentation should be applied to different NLP applications.

Chapter 7

Conclusions

This chapter briefly reviews the thesis, points out future work, and gives an outlook for longer term research.

7.1 Thesis Review

We have proposed using statistical n-gram models for language independent text learning. Statistical language modeling has a strong basis in information theory and removes many ad hoc procedures from traditional text learning systems. The n-gram modeling approach also facilitates language independence. In this thesis, we have achieved improvements in three different areas of text learning based on statistical n-gram language modeling and unsupervised machine learning.

- Language independent and task independent text classification

  We proposed using a simple back-off n-gram language model for language and task independent text classification. The approach is a generalization of naive Bayes classification which considers local Markov dependencies, resulting the chain augmented naive Bayes classifier. It effectively copes with the feature explosion problem by incorporating back-off models and sophisticated smoothing techniques. Thus it avoids an explicit feature selection step critical to most traditional text classifiers, while efficiently dealing with unseen features in a better way than traditional text classifiers. The approach can be applied at both the word level and the character level. When applied at the character level, it avoids the explicit word segmentation problems that occur in many Asian languages such as Chinese and Japanese, and thus it allows a completely language independent approach to text classification. Experiments across various languages (Greek, English, Chinese and Japanese) and various text classification tasks (language identification, authorship attribution, text genre classification, topic detection, and sentimental classification) demonstrate the success of this approach.

- Language independent lexical learning and unsupervised word segmentation

  We proposed using an unsupervised approach based on n-gram models for language independent lexical learning. Given such a model, one segments new sequences based on dynamic programming. To reduce the sparse data problem occurring in traditional EM approaches, we propose a hierarchical method, a self-supervised method, and a mutual information based lexicon pruning method. These methods efficiently address the sparse data problem while helping EM overcome local maxima problems. Experiments on artificial English data and real Chinese data show their effectiveness.

- Chinese text retrieval with unsupervised word segmentation

We further applied unsupervised word segmentation to Chinese text retrieval. The dynamic segmentation method based on the automatically learned lexicon combines many of the advantages of traditional word segmentation methods while overcoming their shortcomings. Experiments on the TREC-5 and TREC-6 datasets show the success of this method. We also investigated the relationship between word segmentation and retrieval performance in Chinese information retrieval. Here we found that the relationship is not monotonic as commonly expected. These findings are of theoretical and practical importance to both Chinese word segmentation and Chinese text retrieval.

7.2 Research Outlook

We have demonstrated the success of statistical n-gram language modeling and unsupervised learning in various text learning problems. However, there are still many research questions that need to be addressed in the near future. One important ongoing investigation is to incorporate unlabeled data into the chain augmented Naive Bayes (CAN) model, in a similar fashion to the plain naive Bayes model [91, 101]. The tricky part about incorporating unlabeled data in the CAN model is that we have to consider back-off and smoothing issues. We have solved most of these issues and are currently beginning experiments with unlabeled data.

In this thesis, we have demonstrated the success of statistical language modeling for text learning problems. In fact, statistical language modeling is a general approach for arbitrary sequence learning problems and can be applied in a much broader area than text. Recently we have successfully applied this approach to web usage mining by considering visited objects as a basic unit for language modeling. Here we proposed a dynamic session boundary detection method using entropy evolution based on statistical language modeling. Experiments on association rule learning based on this new approach improves traditional ad hoc session detection significantly [61].

Many other applications can also be formulated as a language modeling problem, such as gene sequence analysis, music analysis, etc. Figure 7.1 gives an outlook of possible research based on statistical language modeling and machine learning. We are interested in the following research problems.

- Improving statistical language modeling itself

  Although language modeling is a hard and well studied problem, it is still worth further investigation due to its importance. N-gram language models are still the most successful model in this area. However, with the development of faster hardware and developments in machine learning, many currently impractical models could have great applicability in the future. Effectively combining n-gram models with other constraint models such as context free grammars and latent semantic indexing remains an important research topic.

- Web mining

  The rapidly expanding Web contains a vast amount of data containing useful information waiting to be discovered. Web usage mining is a recently established area that focuses on developing techniques for discovering usage patterns in Web log data to better serve the needs Web-based applications. One important Web usage mining problem is to learn interesting association rules from Web logs. Language modeling can be successfully applied in many of Web mining applications. Our first experiments on association rule learning, using the language model based session detection approach outline above, has already shown the promise of language modeling in this area [61]. We are currently investigating other applications such as sequential pattern learning.

- Multimedia information retrieval and bio-informatics

  Multimedia information retrieval and bio-informatics are two other information access problems that have been emerging recently. They share many similarities with text learning problems, and many text learning approaches can be readily applied in these fields. For example, unsupervised word segmentation methods can be applied to music segmentation and gene sequence segmentation, and language modeling techniques can be used to compare the similarities between music sequences and gene sequences.

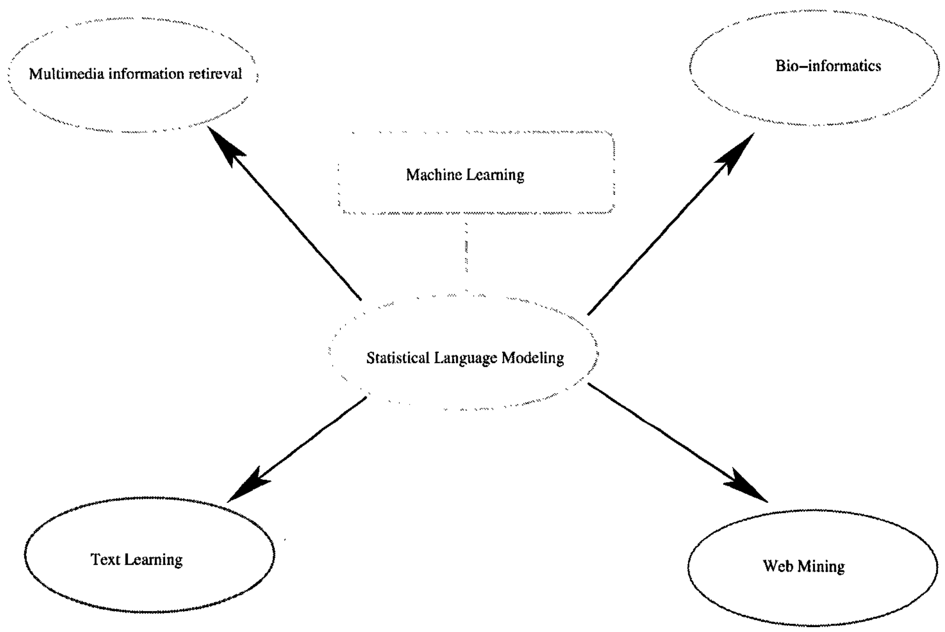
Multimedia information retireval
Bio–informatics
Machine Learning
Statistical Language Modeling
Text Learning
Web Mining

Figure 7.1: Research outlook

Appendix A

Normalization Constant β in the Back-off Model

Let $h = w_{i-n+1}..w_{i-1}$ and $h^{-} = w_{i-n+2}..w_{i-1}$. Then we have the following:

$$\sum_x \Pr(x|h) = \sum_{x:\#(h,x)>0} \Pr(x|h) + \sum_{x:\#(h,x)=0} \Pr(x|h) \tag{A.1}$$

$$= \sum_{x:\#(h,x)>0} \hat{Pr}(x|h) + \sum_{x:\#(h,x)=0} \beta(h) \Pr(x|h^{-}) \tag{A.2}$$

by plugging in Equ. (3.7) to Equ. (A.1).

To ensure that $\Pr(x|h)$ is normalized, we should have

$$\sum_x Pr(x|h) = 1 \tag{A.3}$$

Thus, combining Equ. (A.2) and Equ. (A.3) yields

$$\beta(h) \sum_{x:\#(h,x)=0} Pr(x|h^-) = 1 - \sum_{x:\#(h,x)>0} \hat{P}r(x|h)$$

Hence,

$$\begin{aligned}
\beta(h) &= \frac{1 - \sum_{x:\#(h,x)>0} \hat{P}r(x|h)}{\sum_{x:\#(h,x)=0)} Pr(x|h^-)} \\
&= \frac{1 - \sum_{x:\#(h,x)>0} \hat{P}r(x|h)}{1 - \sum_{x:\#(h,x)>0} Pr(x|h^-)} \\
&= \frac{1 - \sum_{x:\#(h,x)>0} \hat{P}r(x|h)}{1 - \sum_{x:\#(h,x)>0} \hat{P}r(x|h^-)}
\end{aligned}$$

This proves the Equ. (3.9).

Appendix B

An Example Usage of the Waterloo LM Toolkit

In this appendix, we introduce the functionalities of our Waterloo language modeling toolkit, and give an example usage (Section B.3). The use of the toolkit is divided into two phases: generating n-gram language models (Section B.1) and evaluating language models on testing data (Section B.2).

B.1 Generating n-gram language models

Given a training corpus, there are four steps to generate a language model.

1. *text2wfreq*

   The first step is to generate all word frequencies from the training corpus.

Command syntax:

$text2wfreq$ [$-hashsize$ (size of hash table)]

$-$ in (required training file)

[$-out$ (output file, default to screen)]

[$-unit$] (0 for word level;

1 for English character level;

2 for Chinese character level;

3 for Japanese character level, default set to 0)

More about the parameters:

$-in$: specifies the training corpus. This parameter is a mandatory parameter.

$-out$: specifies the output of word frequency file. If no output file is specified, it outputs to the screen.

$-unit$: specifies the basic unit used in the language model. It could be the English word level (0), English character level (1), Chinese character level (2), or Japanese character level (3). The default is set to be English word level (0).

$-hashsize$: Higher values for the -hash parameter require more memory, but can reduce computation time. Normally, one does not need to set this.

2. $wfreq2vocab$

   The second step is generating a vocabulary from the word frequencies generated in step 1.

Command syntax:

```
wfreq2vocab [−hashsize (size of hash table)]
            − in (wfreq file)
            − out (output vocab file)
            [−top (select top N words only, default 20000)]
```

More about the parameters:

−in: specifies the word frequency file generated by command text2wfreq.

−out: specifies the vocabulary file name.

−top: specifies how many of the most frequent words are to be selected from the word frequency file. Default is set to be 20000.

3. *text2idngram*

   The third step is to generate the n-grams from the training corpus. The n-grams are represented with integer word index numbers.

Command syntax:

text2idngram [$-n$ (ngram order, default 3)]

$-$ *vocab* (vocab file)

$-$ *in* (training file)

[$-$*out* (output file, default to screen)]

[$-$*onlyflag*]

[$-$*bwritewords* (1 or 0, default 0)]

[$-$*independence* (1 or 0, default 0)]

[$-$*ex_unk* default is not set]

[$-$*unit*] (0 for word level;

1 for English character level;

2 for Chinese character level;

3 for Japanese character level, default set to 0)

More about the parameters:

n: specifies the order of n-gram. The default is set to be 3, which generates 3-grams.

$-$*vocab*: specifies the vocabulary file generated by wfreq2vocab command.

$-$*in*: specifies the training corpus. It is the same parameter as that used in command text2wfreq.

$-$*out*: specifies the file name for the n-grams. For example, if the file name is

"NGram" and the order is 3, the 1-grams are stored in file "NGram_1", the 2-grams are stored in file "NGram_2" and the 3-grams are stored in file "NGram_3" respectively.

$-onlyflag$: If this flag is set, only the specified n-grams are generated, otherwise all n-grams from order 1 to n are generated. The default is all.

$-bwritewords$: If this is set to be 1, then the words are also written into the n-gram files. Otherwise only the indices are written. The default is that no words are written.

−independence: If this is set. The n-gram language model will assume independence between sentences. Sentences are split by punctuation marks, including period(.), question mark (?) and !. The default is to assume dependence.

−ex_unk: If this is set, the n-gram model will exclude out of vocabulary n-grams in the training corpus. This happens when the selected vocabulary does not contain all the words occurring in the training corpus. The default is to include out of vocabulary n-grams.

4. *idngralm2lm*

The fourth and last step is to generate a n-gram language model from the n-gram files.

Command syntax:

$idngram2lm\ -vocab$ (vocabfile)

$-idngram$ (idngramfile)

$[-n$ (ngram order)$]$

$[-out$ (output file)$]$

$[-vocab\_type$ (vocabulary type)$]$

$[-lin\_abs| - linear| - witten\_bell| - absolute| - good\_turing|addone]$

More about the parameters:

$-vocab$: specifies the vocabulary, which is generated by command wfreq2vocab.

$-idngram$: specifies the n-gram file name. This is the same parameter as that used in text2idngram command.

$-n$: sets the order of n-gram language model. The default is set to be 3.

$-out$: specifies the language model file name.

$-vocab\_type$: specifies the vocabulary type. 1 indicates an open vocabulary, which include a UNK word for unseen words, and 0 indicates a closed vocabulary, which does not allow unseen words.

$[-lin\_abs| - linear| - witten\_bell| - absolute| - good\_turing|addone]$: specifies the smoothing technique. The default is linear smoothing.

The above command syntax information can be displayed by typing "$-help$" following any command.

B.2 Evaluating language models on test data

After a language model is generated, it could be used to evaluate the perplexity of a test document. There is only one command for evaluation.

Command syntax:

evallm $-$ *lm* (* required, language modeling file)

$-$ *vocab* (* required, vocabulary file)

[$-$*independence*] (sentence independence assumption ?)

[$-$*unit*] (0 for word level;

1 for English character level;

2 for Chinese character level;

3 for Japanese character level, default set to 0)

[$-$*test* < *file* >]

[$-$*include_unks*]

More about the parameters:

$-$*lm*: specifies a language model to be used.

$-$*vocab*: specifies the vocabulary, which is generated by command wfreq2vocab.

[$-$*independence*]: If this is set, the evaluation assumes that each sentence is independence.

[$-$*unit*]: This is the same parameter as that used in commands text2wfreq and text2idngram.

[$-$*test*]: This option allows for batch evaluation. One can put all your file NAMES with a full path into a file, where each line is a file name to be evaluated. The program will then evaluate all of these files directly. Otherwise, the program will enter an interactive evaluation environment where one can manually specify files.

[*−include_unks*]: If this flag is set on, the perplexity calculation will include OOV words. Otherwise, the OOV words are ignored for perplexity evaluation.

Interactive Evaluation:

After the "evallm" program enters an interactive evaluation program (by not specifying the -test parameter at command line), there are three commands for interactive operations.

1. help

   This command displays a help menu designed to help users select appropriate commands.

   Command syntax:

   help

2. perplexity

   This command computes the perplexity/entropy of a testing document.

Command syntax:

$$\textit{perplexity} - \textit{test} \quad \text{testfile}$$
$$[-\textit{include\_unks}]$$

More about the parameters:

$-$*test*: specifies the file to be tested.

$-$*include_unks*: This is optional. If it is set, the perplexity computation will include OOV words.

3. validate

   This command validates the sum of the probabilities of all the words given a context is 1.

   Command syntax:

   $$\textit{validate} \quad \text{context}$$

   More about the parameters:

   The context specifies the words which will be conditioned on.

4. quit

   This command exits the program.

Command syntax:

quit

B.3 An Example Usage

We now show how the toolkit works through a concrete example. Here 80% of the Brown corpus is randomly selected for training and the remaining 20% is used for testing. The data is available under the toolkit directory "data/English".

Generating a language model:

1. *textwfreq*

   Input:

   ```
   text2wfreq -in ../data/English/brown_corpus_0.8_train -out wfreq -unit 0
   ```

   Output:

   ```
   text2wfreq: 1000000 ../data/English/brown_corpus_0.8_train wfreq 0

   Total Units Num: 911259

   text2wfreq:done
   ```

   Explanation:

   The program takes the file "../data/English/brown_corpus_0.8_train" as a training corpus and generates 911259 units in total. The basic unit here is a word. The output is written to the file "wfreq" in the current directory.

2. *wfreq2vocab*

Input:

```
wfreq2vocab –in wfreq –out vocab –top 15000
```

Output:

```
reading the words from wfreq file ...done!
sorting counts ... done!
sorting alphabetically...done!
Generate a vocabulary vocab containing the most frequent 15000 words from wfreq It accounts for
0.9608 = (875517/911259) word occurrence!
```

Explanation:

The program takes the word frequency file "wfreq" and generate a vocabulary file "vocab" which contains the 15000 most frequent words. These words account for 96.08% of all word occurrences in the training corpus (875517 out of 911259).

3. *text2idngram*

Input:

```
text2idngram –n 3  –vocab vocab –in ../data/English/brown_corpus_0.8_train  –out NGram
-ex_unk 1
```

Output:

```
text2idngram: -n 3 -vocab vocab -in ../data/English/brown_corpus_0.8_train -out NGram -onlyflag
0 -bwritewords=0-independence 0 -ex_unk 1 -unit 0
Vocabulary Size : 15000
getting 1 grams ....
write to file:NGram...
read 1-Gram into memory from file: NGram_1...
totalWords:15000
sorting...
output to NGram_1...
total 1 grams(uniqueGrams) 875517(15000)
getting 2 grams ....
write to file:NGram...
read 2-Gram into memory from file: NGram_2...
totalWords:289799
sorting...
output to NGram_2...
total 2 grams(uniqueGrams) 841706(280616)
getting 3 grams ....
write to file:NGram...
read 3-Gram into memory from file: NGram_3...
totalWords:603079
sorting...
```

```
output to NGram_3...

total 3 grams(uniqueGrams) 810291(603079)

---------------- results ----------------

1-grams: 875517(15000)

2-grams: 841706(280616)

3-grams: 810291(603079)

text2idngram:done totalGrams: 2527514 totalUniqueGrams:898695

----------------------------------------
```

Explanation:

The programs generates all of the 1-grams, 2-grams, and 3-grams from the training corpus. The n-grams are stored into files "NGram_1", "NGram_2", and "NGram_3" respectively. A total of 875517 uni-grams (the length of training corpus) were extracted, of which 15000 are distinct. Similarly, 841706 bi-grams were extracted, of which 289799 are distinct. and 810291 3-grams were extracted, of which 658698 are distinct.

4. *idngram2lm*

Input:

```
idngram2lm -vocab vocab -idngram NGram -out LIN3.arpa -n 3 -linear
```

Output:

```
This is a 3-gram, back_off=1, idngramFile=NGram szVocFile=vocab

...
(Detailed information is omitted here)
...

————————— discounting ratios ——————————————
Linear discounting ratios...
1-gram : 0.998714
2-gram : 0.779318
3-gram : 0.394262
——————————————————————————————————————

...
(Detailed information is omitted here)
...

write language model to file LIN3.arpa...
Bo_Weight memory is freed
Unigram prob and log_prob memory is freed
```

Explanation:

The program generates a tri-gram language model using linear smoothing. The smoothing constants for the uni-grams, bi-grams, and tri-grams are 0.998714, 0.779318, and 0.394262 respectively. The language model is written into the file "LIN3.arpa".

Now that a language model has been generated, it can be used to evaluate the perplexity of an unseen testing document.

Evaluating language models on testing data:

1. evallm

   Input:

   ```
   evallm -vocab vocab -lm LIN3.arpa
   ```

   This command loads the language model "LIN3.arpa" and moves the program into an interactive environment.

   Interactive Commands:

   (a) perplexity

      Input:

      ```
      perplexity -test ../data/English/brown_corpus_0.2_test
      ```

      Output:

```
Perplexity/Entropy = 432.065562/8.755106 dUNKProb = 0.000000 dSum_Log_Prob=-632352.699000
totalWords = 259749 OOV words #=19817 (0.076293) totalUsed Words = 239932
3-gram hit number :58641 (0.244407)
2-gram hit number :102848 (0.428655)
1-gram hit number :78443 (0.326938)
Character_perpelxity/entropy = 4.477070/2.162555 Character_number = 971365

Missing words: missing#/(missing#+hit#)
3-gram: 181289/(181289+58641) = (0.755591)
2-gram: 78442/(78442+102848) = (0.432688)
1-gram: 0/(0+78443) = (0.000000)
```

Explanation:

This command evaluates the perplexity of testing document "../data/English/brown_corpus_0.2_test"

The perplexity and entropy of the testing document are 432.065562 and 8.755106 respectively. Additional information about how many n-grams were observed and how many were missing were also printed out.

(b) validate

Input:

```
validate see
```

Output:

```
prob(* | see ) = 1.000016
```

Explanation:

The commands computes the sum of all conditional probabilities given the context *see.* The sum of the probabilities is 1.000016, which is correct up to round off error. (The value is supposed to be 1.)

References

[1] A. Aizawa. Linguistic Techniques to Improve the Performance of Automatic Text Categorization. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposim (NLPRS)*, pages 307–314, 2001.

[2] R. Ando and L. Lee. Mostly-Unsupervised Statistical Segmentation of Japanese: Application to Kanji. In *First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 241–248, 2000.

[3] R. Ando and L. Lee. Mostly-Unsupervised Statistical Segmentation of Japanese Kanji Sequences. *Journal of Natural Language Engineering*, 9(1), 2003.

[4] L. Bahl, F. Jelinek, and R. Mercer. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.

[5] D. Baker, T. Hofmann, A. McCallum, and Y. Yang. A Hierarchical Probabilistic Model for Novelty Detection in Text. CMU, Unpublished manual.

[6] M. Beaulieu, M. Gatford, X. Huang, S. Robertson, S. Walker, and P. Williams. Okapi at TREC-5. In *Proceedings of TREC-5*, pages 143–166, 1997.

[7] T. Bell, J. Cleary, and I. Witten. *Text Compression.* Prentice Hall, 1990.

[8] J. Bellegarda. Exploiting Latent Semantic Information in Statistical Language Modeling. *Proceedings of the IEEE*, 88(8):1279–1296, 2000.

[9] D. Benedetto, E. Caglioti, and V. Loreto. Language Trees and Zipping. *Physical Review Letters*, 88, 2002.

[10] Y. Bengio, R. Ducharme, and P. Vincent. A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems 13 (NIPS)*, 2001.

[11] A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, 1998.

[12] M. Brand. Structure Learning in Conditional Probability Models Via An Entropic Prior And Parameter Extinction. *Neural Computation*, 11:1155–1182, 1999.

[13] M. Brent. An Efficient, Probabilistically Sound Algorithm for Segmentation and Word Discovery. *Machine Learning*, 34:71–106, 1999.

[14] M. Brent and T. Cartwright. Distributional Regularity and Phonotactics Are Useful for Segmentation. *Cognition*, 61:93–125, 1996.

[15] M. Brent and X. Tao. Chinese Text Segmentation With MBDP-1: Making the Most of Training Corpora. In *Proceedings of 39th Annual Meeting of Association for Computational Linguistic (ACL)*, France, 2001.

[16] P. Brown, Della P., P. deSouza, J. Lai, and R. Mercer. Class-based n-gram Models of Natural Language. *Computational Linguistics*, 18:467–479, 1992.

[17] C. Buckley, A. Singhal, and M. Mitra. Using Query Zoning and Correlation within SMART: TREC-5. In *Proceedings of TREC-5*, pages 105–118, 1997.

[18] C. Buckley, J. Walz, M. Mitra, and C. Cardie. Using Clustering and SuperConcepts Within SMART: TREC-6. In *Proceedings of TREC-6*, pages 107–124, 1998.

[19] C. Burges. A Tutorial On Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[20] W. B. Cavnar and J. M. Trenkle. N-Gram-Based Text Categorization. In *3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, 1994.

[21] J. S. Chang and K. Y. Su. An Unsupervised Iterative Method for Chinese New Lexicon Extraction. *International Journal of Computational Linguistics & Chinese Language Processing*, 1997.

[22] E. Charniak. Immediate Head Parsing for Language Models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2001.

[23] C. Chelba and F. Jelinek. Exploiting syntactic structure for language modelling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics(ACL/COLING)*, pages 225–231, Montreal, 1998.

[24] A. Chen, J. He, L. Xu, F. Gey, and J. Meggs. Chinese Text Retrieval Without Using a Dictionary. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 42–49. ACM, 1997.

[25] S. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[26] L.F. Chien, T.I. Huang, and M.C. Chien. Pat-tree-based Keyword Extraction for Chinese Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–58. ACM, 1997.

[27] M. Christiansen and J. Allen. Coping with Variation in Speech Segmentation. In A. Sorace, C. Heycock, and R. Shillcock, editors, *Proceedings of GALA 1997: Language Acquisition: Knowledge Representation and Processing*, pages 327–332, University of Edinburgh, 1997.

[28] M. Christiansen, J. Allen, and M. Seidenberg. Learning to Segment Speech Using Multiple Cues: A Connectionist Model. *Language and Cognitive Processes*, 13:221–268, 1998.

[29] P. Clarkson. *Adaptation of Statistical Language Models for Automatic Speech Recognition.* PhD thesis, Cambridge University Engineering Department, 1999.

[30] P.R. Clarkson and R. Rosenfeld. Statistical Language Modeling Using the CMU-Cambridge Toolkit From Proceedings. In *ESCA Eurospeech*, 1997.

[31] J. Cleary and I. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transaction on Communications*, 32(4):396–402, 1984.

[32] M. Collins. EM (Expectation Maximization) Algorithm. In *A review of EM*, University of Pennsylvania, 1997.

[33] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 1991.

[34] D. Dahan and M. Brent. On the discovery of novel word-like units from utterances: An artificial-language study with implications for native-language acquisition. *Journal of Experimental Psychology: General*, 128:165–185, 1999.

[35] M. Damashek. Gauging Similarity with N-Grams: Language-Independent Categorization of Text? *Science*, 267:843 – 848, 1995.

[36] C. de Marcken. The Unsupervised Acquisition of a Lexicon from Continuous Speech. Technical Report AI Memo No. 1558, MIT, Cambridge, Massachusetts, 1995.

[37] S. Deligne and F. Bimbot. Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1995.

[38] A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from Incomplete Data via the EM algorithm. *J. Royal Statist. Soc. Ser.*, B(39), 1977.

[39] P. Domingos and M. Pazzani. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Machine Learning*, 29:103–130, 1997.

[40] J. Stephen Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-Grams as Text.* PhD thesis, University of Western Ontario, London, Ontario, 1999.

[41] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[42] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive Learning Algorithms and Representations for Text Categorization. In *Proceedings of ACM Seventh International Conference on Information and Knowledge Management (CIKM98)*, 1998.

[43] D. Elworthy. Does Baum-Welch Re-estimation Help Taggers? In *Proceedings of the 4th Conference on Applied Natural Language Processing (ANLP)*, Stuttgart, 1994.

[44] S. Eyheramendy, D. Lewis, and D. Madigan. On the Naive Bayes Model for Text Categorization. In *Proceedings Artificial Intelligence & Statistics 2003*, 2003.

[45] S. Foo and H. Li. Chinese Word Segmentation Accuracy and Its Effects on Information Retrieval. *TEXT Technology*, 2001.

[46] D. Freitag and A. McCallum. Information Extraction with HMMs and Shrinkage. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, 1999.

[47] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29:31–163, 1997.

[48] P. Fung. Extracting Key Terms from Chinese and Japnese Text. *International Journal on Computer Processing of Oriental Language*, Special Issue on Information Retrieval on Oriental Languages:99–121, 1998.

[49] X. Ge, W. Pratt, and P. Smyth. Discovering Chinese Words from Unsegmented Text. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 271–272, 1999.

[50] F. Gey, A. Chen, J. He, L. Xu, and J. Meggs. Term importance, boolean conjunct training, negative terms, and foreign language retrieval: probabilistic algorithms at TREC-5. In *Proceedings of the Fifth Text REtrieval Conference*, pages 181–190. NIST special publication, 1996.

[51] J. Goldsmith. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 2001.

[52] J. Goodman. A Bit of Progress in Language Modeling. Technical report, Microsoft Research, 2000.

[53] J. Goodman. Comment on Language Trees and Zipping. Unpublished Manuscript, 2002.

[54] Z. Harris. From phoneme to morpheme. *Language*, 31:190–222, 1955.

[55] J. He, A. Tan, and C. Tan. A Comparative Study on Chinese Text Categorization Methods. In *Proceedings of PRICAI'2000 International Workshop on Text and Web Mining*, pages 24–35, 2000.

[56] J. He, A. Tan, and C. Tan. On Machine Learning Methods for Chinese Documents Classification. *Applied Intelligence*, Special Issue on Text and Web Mining, 2001.

[57] J. Hockenmaier and Brew C. Error driven segmentation of Chinese. *Communications of COLIPS*, 8(1):69–84, 1998.

[58] T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 1999.

[59] D. Holmes and R. Forsyth. The Federalist Revisited: New Directions in Authorship Attribution. *Literary and Linguistic Computing*, 10:111–127, 1995.

[60] Y. Hua. Unsupervised Word Induction Using MDL Criterion. In *Proceedings International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Beijing, 2000.

[61] X. Huang, F. Peng, A. An, D. Schuurmans, and N. Cercone. Session Boundary Detection for Association Rule Learning Using N-Gram Language Models. In *Proceedings of The Sixteenth Candian Conference on Artificial Intelligence (CAI)*, 2003.

[62] X. Huang, F. Peng, D. Schuurmans, N. Cercone, and S. Robertson. Applying Machine Learning for Text Segmentation in Information Retrieval. *Information Retrieval Journal*, To appear, 2003.

[63] X. Huang and S. Roberton. Okapi Chinese Text Retrieval Experiments at TREC-6. In *Proceedings of TREC-6*, pages 137–142, 1998.

[64] X. Huang and S. Robertson. A Probabilistic Approach to Chinese Information Retrieval: Theory and Experiments. In *Proceedings of the BCS-IRSG 2000: the 22nd Annual Colloquium on Information Retrieval Research*, Cambridge, England, 2000.

[65] Stephen Huffman. Acquaintance: Language-Independent Document Categorization by N-grams. In *In Donna K. Harman and Ellen M. Voorhees, editors, Proceedings of TREC-4, 4th Text Retrieval Conference*, pages 359 - 371, 1995.

[66] Y. Ivanov, B. Blumberg, and A. Pentland. Expectation Maximization for Weakly

Labeled Data. In *Proceedings of The Eighteenth International Conference on Machine Learning (ICML)*, 2001.

[67] R. Iyer and M. Ostendorf. Modeling Long Distance Dependence in Language: Topic Mixtures versus Dynamic Cache Models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39., 1999.

[68] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. A Dynamic LM for Speech Recognition. In *Proceedings ARPA Workshop on Speech and Natural Language*, pages 293–295, 1991.

[69] W. Jin. Chinese Segmentation and its Disambiguation. In *MCCS-92-227*, Computing Research Laboratory, New Mexico State University, Las Cruces, New Mexico, 1992.

[70] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, 1998.

[71] S. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):400–401, March 1987.

[72] E. Keogh and M. Pazzanni. A Comparison of Distribution-based and Classification-based Approaches. In *Proceedings Artificial Intelligence & Statistics 1999*, 1999.

[73] B. Kessler, G. Nunberg, and H. Schüze. Automatic Detection of Text Genre. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics (ACL)*, 1997.

[74] S. Khudanpur and J. Wu. Maximum Entropy Techniques for Exploiting Syntactic, Semantic and Collocational Dependencies in Language Modeling. *Computer Speech and Language*, pages 355–372, 2000.

[75] C. Kit. *Unsupervised Lexical Learning as Inductive Inference.* PhD thesis, University of Sheffield, UK, 2000.

[76] C. Kit and Y. Wilks. Unsupervised Learning of Word Boundary with Description Length Gain. In *Proceedings of Computational Natural Language Learning (CoNLL)*, Bergen, 1999.

[77] K. L. Kwok. Comparing Representations in Chinese Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 34–41. ACM, 1997.

[78] K. L. Kwok. Employing Multiple Representations for Chinese Information Retrieval. *Journal of the American Society for Information Science (JASIS)*, 50(8):709–723, 1999.

[79] K. L. Kwok. Improving English and Chinese Ad-Hoc Retrieval: A Tipster Text Phase 3 Project Report. *Information Retrieval*, 3(4):313–338, 2000.

[80] K. L. Kwok and L. Grunfeld. TREC-5 English and Chinese Retrieval Experiments using PIRCS. In *Proceedings of TREC-5*, 1996.

[81] J. Lafferty, D. Sleator, and D. Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, Cambridge, MA, 1992.

[82] J. Lafferty and C. Zhai. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *24th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[83] K. Lang. Newsweeder: Learning to Filter Netnews. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

[84] Y. Lee and S. Myaeng. Text Genre Classification with Genre-Revealing and Subject-Revealing Features. In *Proceedings of The 25th Annual International ACM SIGIR Conference on Research and Development in Information (SIGIR)*, 2002.

[85] D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, Computer Science Deptment, Univ. of Massachusetts, 1992.

[86] D. Lewis. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *Proceedings 10th European Conference on Machine Learning (ECML)*, 1998.

[87] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts, 1999.

[88] Carla Marceau. Characterizing the Behavior of a Program Using Multiple-Length N-grams. In *In Proceedings of the 2000 new security paradigm workshop*, pages 101–110, 2000.

[89] T. McArthur. *The Oxford Companion to the English Language.* Oxford University Press, 1992.

[90] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Proceedings of AAAI-98 Workshop on "Learning for Text Categorization*, 1998.

[91] A. McCallum and K. Nigam. Employing EM in Pool-Based Active Learning for Text Classification. In *Proceedings of The Fifteenth International Conference on Machine Learning (ICML)*, 1998.

[92] Arthur Nadas. On Turing's Formula for Word Probabilities. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(6):1414–1416, 1985.

[93] Arthur Nadas. Good, Jelinek, Mercer, and Robins on Turing's estimate of probabilities. *American Journal of Mathematical and Management Sciences*, 11:229–308, 1991.

[94] M. Nechyba. The Expectation-Maximization(EM) algorithm. EEL6935: Machine Learning in Robotics II, Spring 2000 Lecture Notes, University of Florida.

[95] C. Nevill-Manning and I. Witten. Compression and Explanation Using Hierarchical Grammars. *Computer Journal*, 40:103–116, 1997.

[96] H. Ney, U. Essen, and Kneser R. On Structuring Probabilistic Dependencies in Stochastic Language Modeling. *Computer Speech and Language*, 8(1):1–28, 1994.

[97] H. Ney, U. Essen, and Kneser R. On the Estimation of 'Small' Probabilities by Leaving-One-Out. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(12):1202–1212, 1995.

[98] J.Y. Nie, J. Gao, J. Zhang, , and M. Zhou. On the use of words and n-grams for chinese information retrieval. In *In Proceedings of the Fifth International Workshop on Information Retrieval with Asian Languages (IRAL)*, Hong kong, 2000.

[99] J.Y. Nie and F. Ren. Chinese Information Retrieval: Using Characters or Words? *Information Processing and Management*, 35:443–462, 1999.

[100] J.Y. Nie, X. Ren, and M. Brisebois. On Chinese Text Retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 225–233. ACM, 1996.

[101] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents Using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[102] D. Palmer and J. Burger. Chinese Word Segmentation and Information Retrieval. In *AAAI Spring Symposium on Cross-Language Text and Speech Retrieval, Electronic Working Notes*, 1997.

[103] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[104] F. Peng, X. Huang, D. Schuurmans, and N. Cercone. Investigating the Relationship of Word Segmentation Performance and Retrieval Performance in Chinese IR. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 793–799, Taipei, Taiwan, 2002.

[105] F. Peng, X. Huang, D. Schuurmans, N. Cercone, and S. Robertson. Using Self-supervised Word Segmentation for Chinese Information Retrieval. In *Proceedings of the 25th ACM SIGIR Annual Conference (SIGIR)*, Tampere, Finland, 2002.

[106] F. Peng and D. Schuurmans. A Hierarchical EM Approach to Word Segmentation. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NL-PRS)*, pages 475–480, Tokyo, Japan, 2001.

[107] F. Peng and D. Schuurmans. Self-supervised Chinese Word Segmentation. In *F. Hoffman et al. (Eds.): Advances in Intelligent Data Analysis, Proceedings of the Fourth International Conference (IDA-01), LNCS 2189*, pages 238–247, Cascais, Portugal, 2001. Springer-Verlag Berlin Heidelberg.

[108] F. Peng and D. Schuurmans. Combining Naive Bayes and N-Gram Language Models for Text Classification. In *Proceedings of The 25th European Conference on Information Retrieval Research (ECIR)*, Pisa, Italy, 2003.

[109] F. Peng, D. Schuurmans, V. Keselj, and S. Wang. Language Independent Author Attribution with N-Gram Language Models. In *Proceedings of the 10th European Chapter of Association of Computational Linguistics (EACL)*, Budapest, Hungary, 2003.

[110] F. Peng, D. Schuurmans, and S. Wang. Augumenting Naive Bayes Text Classifier with Statistical Language Models. *Information Retrieval Journal*, invited submssion to special issue of ECIR-2003, 2003.

[111] F. Peng, D. Schuurmans, and S. Wang. Language and Task Independent Text Cate-

gorization with Simple Language Models. In *Proceedings of Human Language Technology Conference and North America Chapter of Asociation of Computational Linguistics (HLT-NAACL)*, Edmonton, Canada, 2003.

[112] J. Ponte and W. Croft. Useg: A Retargetable Word Segmentation Procedure for Information Retrieval. In *Symposium on Document Analysis and Information Retrieval 96 (SDAIR)*, 1996.

[113] J. Ponte and W. Croft. A Language Modeling Approach to Information Retrieval. In *Proceedings of ACM Research and Development in Information Retrieval (SIGIR)*, pages 275–281, 1998.

[114] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of IEEE*, 77(2), 1989.

[115] J. Rennie. Improving Multi-class Text Classification with Naive Bayes. Master's thesis, M.I.T., 2001.

[116] I. Rish. An Empirical Study of the Naive Bayes Classifier. In *Proceedings of IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence.*, 2001.

[117] S. Robertson and K. Sparck Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[118] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 232–241, 1994.

[119] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Statistical Approach.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1994.

[120] R. Rosenfeld. Two decades of Statistical Language Modeling: Where Do We Go From Here? *Proceedings of the IEEE*, 88:1270–1278, August 2000.

[121] M. Sahami. *Using Machine Learning to Improve Information Access.* PhD thesis, Stanford University, Computer Science Department, 1998.

[122] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[123] G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[124] J. C. Schmitt. Trigram-based Method of Language Identification. October 1991. U.S. Patent No. 5,062,143.

[125] D. Schuurmans. A new metric-based approach to model selection. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI)*, 1997.

[126] S. Scott and S. Matwin. Feature Engineering for Text Classification. In *Proceedings of The Sixteenth International Conference on Machine Learning (ICML)*, pages 379–388., 1999.

[127] F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[128] H. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

[129] M. Siu and M. Ostendorf. Variable N-gram Language Modeling and Extensions for Conversational Speech. *IEEE Transactions on Speech and Audio Processing*, 8:63–75, 2000.

[130] N. Slonim and N Tishby. Document Clustering using Word Clusters via the Information Bottleneck Method. In *Proceedings of 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR)*, 2000.

[131] K. Sparck-Jones. Search Relevance Weighting Given Little Relevance Information. *Journal of Documentation*, 35(1), 1979.

[132] K. Sparck-Jones. The Role of Artificial Intelligence in Information Retrieval. *Journal of the American Society for Information Science*, 42(8):558–565, 1991.

[133] R. Sproat and C. Shih. A Statistical Method for Finding Word boundaries in Chinese Text. *Computer Processing of Chinese and Oriental Language*, 4:336–351, 1990.

[134] R. Sproat, C. Shih, W. Gale, and N. Chang. A Stochastic Finite-state Word-segmentation Algorithm for Chinese. *Computational Linguistics*, 22 (3):377–404, 1996.

[135] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Automatic Text Categorization in Terms of Genre and Author. *Computational Linguistics*, 26(4):471–495, 2000.

[136] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Computer-based Authorship Attribution without Lexical Measures. *Computers and the Humanities*, 35:193–214, 2001.

[137] Daniel R. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms.* PhD thesis, Leiden University, 2002.

[138] W. Teahan and D. Harper. Using Compression-Based Language Models for Text Categorization. In *Proceedings of Workshop on Language Modeling and Information Retrieval (LMIR)*, 2001.

[139] W. J. Teahan, Y. Wen, R. McNab, and Witten I. H. A Compression-based Algorithm for Chinese Word Segmentation. *Computational Linguistics*, 26(3):375–393, 2001.

[140] D. Tkach. Text Mining Technology: Turning Information into Knowledge. A white paper from IBM, 1997.

[141] S. Tong. *Active Learning: Theory and Applications.* PhD thesis, Stanford University, 2001.

[142] P. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[143] A. Turpin and A. Moffat. Statistical Phrases for Vector-Space Information Retrieval. In *22th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[144] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[145] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Proceedings of the Sxth Text REtrieval Conference (TREC-6).* NIST Special Publication, 1998.

[146] S. Wang, D. Schuurmans, and F. Peng. Latent Maximum Entropy Approach for Semantic N-gram Language Modeling. In *Proceedings of Ninth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2003.

[147] S. Wang, D. Schuurmans, F. Peng, and Y. Zhao. Semantic N-gram Language Modeling with The Latent Maximum Entropy Principle. In *Proceedings of The 28th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

[148] R. Wilkinson. Chinese Document Retrieval at TREC-6. In *Proceedings of TREC-6*, 1998.

[149] I. Witten and T. Bell. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory 37(4)*, 37(4), 1991.

[150] Y. Wu, Q. Tian, and T. Huang. Discriminant-EM Algorithm with Application to Image Retrieval. In *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume I, pages 222–227, Hilton Head Island, SC, 2000.

[151] Z. Wu and G. Tseng. Chinese Text Segmentation for Text Retrieval: Achievements and Problems. *Journal of the American Society for Information Science (JASIS)*, 44(9):532–542, 1993.

[152] Y. Yang. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 1(1/2):67–88, 1999.

[153] E. J. Yannakoudakis and P. J. Hutton. An Assessment of N-Phoneme Statistics in Phoneme Guessing Algorithms Which Aim to Incorporate Phonotactic Constraints. *Speech Communications*, 11(6):581–602, 1995.

[154] J. Zhao, J. Gao, E. Chang, and M. Li. Lexicon Optimization for Chinese language modeling. In *Proceedings International Symposium Conference on Spoken Language Processing (ISCSLP)*, Beijing, 2000.

[155] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Cambridge, MA, 1949.