Learning Features for Text Classification

Bin Zhang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2013

Reading Committee:

Mari Ostendorf, Chair

Emily M. Bender

Maryam Fazel

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Learning Features for Text Classification

Bin Zhang

Chair of the Supervisory Committee:
Professor Mari Ostendorf
Electrical Engineering

Text classification is a general and important machine learning problem. For example, topic classification of text documents has been extensively studied for more than a decade, and simple word features are found to be very indicative of topics. Researchers have been focusing mostly on machine learning of classifiers instead of that of features. More recently, classification of sentiment, agreement and opinions in social media has drawn much attention, where individual word features are no longer sufficiently discriminative. Because good features are important to these tasks, engineering features becomes a crucial step in developing good text classification systems. However, feature engineering involves much manual work and is time-consuming.

Another challenge to many text classification tasks is limited labeled training data. Large amounts of unlabeled data are available but they are often not used in supervised classifier training. A big issue related to features caused by limited labeled data is that only limited features are seen, and classifiers trained by supervised learning cannot use features that are unseen in training data.

This thesis attempts to address both issues by applying machine learning to text features. A type of feature, i.e., phrase patterns, and the efficient algorithm to learn them from labeled training data, are proposed. Phrase pattern features are particularly useful for tasks involving modeling long-range complex behaviors as we see in social media data, and they are more flexible than n-gram features. The learned phrase patterns can contain both words

and word classes, which improves generalizability. Significant performance improvements are observed in multiple conversational text classification tasks.

This thesis also proposes feature affinity and cluster regularization, which uses feature relationships learned from unlabeled data to regularize training. This regularization scheme converts supervised learning text classification to semi-supervised, and it is simple to control the relative importance of the knowledge learned from unlabeled data. Using this method, features that are unseen in the labeled data get non-zero weights due to their relationships to seen features. These algorithms are evaluated in topic and sentiment classification tasks, achieving significant improvements. This thesis also studies the problem of learning feature relationships for high-dimensional features using a mixture co-occurrence model, making our approach applicable to complex task classification tasks where large amounts of unlabeled data are available. Multiple conversational text classification tasks are studied in our experiments, and significant performance improvements are demonstrated.

TABLE OF CONTENTS

Page

List of Figures iii

Glossary iv

Chapter 1: Introduction 1
1.1 Text Classification as a General Problem 2
1.2 Text Features 3
1.3 Feature Engineering vs. Feature Learning 4
1.4 Contributions 5
1.5 Dissertation Overview 7

Chapter 2: Background 9
2.1 Features for Text Classification 9
2.2 Feature Processing Techniques 13
2.3 Maximum Entropy Models 17
2.4 Regularization of Maximum Entropy Models 18
2.5 Semi-supervised Maximum Entropy Models 19
2.6 Summary 21

Chapter 3: Learning Phrase Patterns 22
3.1 Phrase Patterns and Word Classes 23
3.2 Learning Discriminative Phrase Patterns 24
3.3 Experiments 36
3.4 Summary 47

Chapter 4: Leveraging Feature Relationships Learned from Unlabeled Data 49
4.1 Feature Affinity Regularization 51
4.2 Feature Cluster Regularization 57
4.3 Fixed Point Weight Propagation 62
4.4 Learning Feature Affinity 66

4.5 Learning Feature Clustering 70
4.6 Experiments 72
4.7 Summary 86

Chapter 5: Modeling High-Dimensional Feature Relationships 88
5.1 Co-occurrence Models 90
5.2 Experiments on Word Features 98
5.3 Experiments on High-Dimensional Features 103
5.4 Summary 108

Chapter 6: Conclusion 110
6.1 Contributions 110
6.2 Future Directions 113

Bibliography 115

LIST OF FIGURES

Figure Number Page

3.1 Example of a phrase pattern matched by two sentences. We expand the words in sentences by word classes (POS and word polarity shown). Boxed elements are matches to the phrase pattern. 25

4.1 Example feature graphs: (a) an unseen feature is connected to only one seen feature, (b) an unseen feature is connected to two seen features 54

4.2 Weights learned by (a) FAR with no l_2 on feature C, and (b) FAR with l_2 on feature C. Note that D represents the bias term. 55

4.3 Example feature graph with (a) and without (b) FAR 56

4.4 Feature graph: unseen feature U_n is indirectly connected to seen feature S through $n-1$ intermediate unseen features 65

4.5 Movie review sentiment classification results (FAR) 78

4.6 20 Newsgroups topic classification results (FAR) 79

4.7 Reuters-21578 topic classification results (FAR) 80

4.8 Movie review classification results (FCR) 81

4.9 20 Newsgroups topic classification results (FCR) 82

4.10 Reuters-21578 topic classification results (FCR) 83

4.11 Feature weight change with vs. without FPWP in movie review classification. FAR with NCOOC is used. The jumpiness of the curves is caused by the automatic adjustment of step size in L-BFGS. 85

4.12 Training objective function value with vs. without FPWP in movie review classification. FAR with NCOOC is used. 86

5.1 Movie review sentiment classification results with full **W** and mixture co-occurrence models 100

5.2 20 Newsgroups topic classification results with full **W** and mixture co-occurrence models 101

5.3 Reuters-21578 topic classification results with full **W** and mixture co-occurrence models 102

GLOSSARY

20 NEWSGROUPS: A text data set collected from Newsgroups. The documents have been labeled into 20 topic categories.

ALIGNMENT MOVE: an utterance in which an individual attempts to positively or negatively align herself to other individuals by showing agreement and dissent, respectively.

AUTHORITY CLAIM: an utterance through which an individual associates herself with a construct that she believes will be influential in persuading those attending to the interaction that her ideas merit consideration.

ASR: automatic speech recognition.

COOC: co-occurrence probability.

DICE: Dice coefficient.

EM: expectation maximization

FAR: feature affinity regularization.

FCR: feature cluster regularization.

FPWP: fixed point weight propagation.

HYPER-PARAMETER: A parameter that is manually tuned to the optimal value.

JAC: Jaccard index.

L-BFGS: limited-memory BFGS, a quasi-Newton optimization method.

LDA: latent Dirichlet allocation.

MAP-REDUCE: a programming model for processing large data sets over clusters of computers in parallel.

MAXENT: maximum entropy model, also known as (multi-class) logistic regression and log-linear model.

NCOOC: normalized co-occurrence probability.

N-GRAM: a tuple of n consecutive words.

NLP: natural language processing.

NPMI: non-negative point-wise mutual information.

PHRASE PATTERN: a sequence of tokens. When it matches a sentence, the corresponding tokens appear in the order specified by the phrase pattern; gaps between tokens are allowed.

PLSA: probabilistic latent semantic analysis.

POS: part-of-speech.

PREFIXSPAN: prefix-projected sequential pattern mining.

REUTERS-21578: A text data set collected from Reuters newswire. It contains 21578 topically labeled news articles.

SVM: support vector machine.

TF-IDF: term frequency – inverse document frequency.

WIKIPEDIA DISCUSSION: Editable pages for Wikipedia editors to discuss changes to an associated article or project page in Wikipedia, also known as Wikipedia talk.

XR: expectation regularization.

ACKNOWLEDGMENTS

The author would like to thank Professor Mari Ostendorf for her mentoring of this dissertation. Without her advising, this dissertation is impossible.

The author would like to thank Professor Emily M. Bender, Professor Maryam Fazel, and Professor Malcolm Parks for taking time to read and comment on this dissertation. The author would also like to thank former thesis committee member, Professor Jeff Bilmes, for his comments to the thesis proposal.

The author would like to thank current and former members of SSLI, including Alex Marin, Brian Hutchinson, Julie Medero, Sangyun Hahn, Nicole Nichols, Yuzong Liu, Ji He, David Aragon, Wei Wu, Anna Margolis, Xin Lei, Mei-Yuh Hwang, Mei Yang, Jeremy Kahn, Dustin Hillard, and Arindam Mandal for invaluable discussions and collaborations.

The author would like to thank Professor Emily M. Bender, Professor Mark Zachry, Jonathan Morgan, Meghan Oxley, and many annotators for their collaboration in the SCIL project, which created most of the labeled data for experiments in this thesis.

The author would like to thank Wen Wang, Jing Zheng, and other researchers in SRI for providing their ASR system.

The author would like to thank IARPA and DARPA for funding the research projects.

DEDICATION

to Maggie and my parents

Chapter 1

INTRODUCTION

Text classification is a type of natural language processing (NLP), where written text (or speech transcripts) is classified according to predefined categories based upon lexical and possibly formatting features. There has been a great amount of work in this area, especially after easy access to text data was made available by the Internet. Previous text classification research includes topic categorization, genre classification, reading level detection, role classification, and sentiment analysis, to name a few.

For decades, researchers have worked on algorithms to automatically categorize text documents according to their topics, as well as retrieve text documents that are similar in topic to the query. These algorithms usually make decisions based on individual words, as they are good topic indicators. Recently, there has been increased availability of online text and audio due to the popularity of social networking web sites, review web sites, online news and magazines, and online interactive forums, etc. Mining information beyond topic has become more desirable. For example, sentiment analysis of online reviews reveals users' opinions with regard to the products, and detection of agreements and disagreements between speakers/writers clusters them into different opinion groups. Individual words are no longer sufficient for decision making, because, for example, a single word can represent different sentiments or opinions in different contexts. More discriminative signals are desirable for achieving high classification accuracy.

In addition to seeking more discriminative signals, effective usage of unlabeled data is becoming more and more important for new text classification applications. On the one hand, it is expensive to label large amounts of text data for these new applications, making labeled text data scarce. On the other hand, increasing amounts of unlabeled data are becoming available. Although we may not get application-dependent information from unlabeled data, it is useful if we can learn characteristics of the domain from unlabeled data

and apply it to different text classification applications.

1.1 Text Classification as a General Problem

In this thesis, we consider text classification as a general problem, where the common elements across the different applications are feature extraction, which converts text into numerical vectors, and classifier learning, which generates decision boundaries in the feature space. By viewing text classification as a general problem within a machine learning framework, many machine learning technologies are applicable regardless of the actual task to which text classification is applied.

Supervised learning is the most commonly used approach to text classification tasks. A set of labeled training text examples are created by human annotators to train a statistical classifier, which then can be applied to new text samples in order to generate hypothetical class labels without human intervention. High classification accuracy can be achieved if sufficient labeled training data are available and match the test data. Both requirements are sometimes challenging. Because labels have to be created on a task-by-task basis, it is difficult to get sufficient labeled training data for new tasks. Training-test mismatch is also a common source of problems, but it may be alleviated by feature normalization or domain transfer learning methods such as structural correspondence learning (SCL) [Blitzer et al., 2006]. Examples of supervised learning methods include naive Bayes, k-nearest neighbors, maximum entropy models (MaxEnt), and support vector machines (SVM).

When no labeled training data are available, text classification may still be accomplished by unsupervised learning. Unsupervised learning is commonly referred to as clustering, where data samples are grouped into clusters based on their similarities or geometrical properties in the feature space. We will not be able to identify the correspondence between clusters and classes, but sometimes one can use some heuristic rules to determine cluster-class correspondence. It should be pointed out that unsupervised text classification is difficult in general. Since no label information is used for training, there is no guarantee that the resulting clusters would correspond to the particular classes that we are interested in. Features may be tweaked to better reflect target classes, but it is arguable whether resources should just be spent in annotating data, because feature tweaking also involves

human effort and can be time-consuming, too. Despite these disadvantages, unsupervised learning still offers us a way to gain knowledge about the data without any annotation. Examples of unsupervised learning include k-means, graph cut, and spectral clustering.

When both labeled and unlabeled training data are available, we can resort to semi-supervised learning. A statistical classifier is trained, but the trainer makes use of both labeled and unlabeled data. There are many variants of semi-supervised learning (c.f. [Zhu, 2005]), and it remains an active research area. Many semi-supervised learning methods are based on assumptions such as that decision boundaries are likely to be in low density regions, that samples in the same neighborhood are likely to share labels, and/or that data are likely to lie on a manifold. Weakly supervised learning is also a possible approach, if it is too costly to create full labels for training data. One can specify salient features or create partial labels. For example, strongly indicative words can be specified [Mann and McCallum, 2010], or a coarser-level of class labels may be provided. However, these algorithms tend to be more complex than supervised learning, and they have to be tailored to particular applications.

1.2 Text Features

For all the machine learning methods introduced above, it is required that the text is converted into a feature representation, which is typically a vector of numbers. This conversion process is called feature extraction [Guyon and Elisseeff, 2006]. How the features are extracted controls the amount of information that goes into the statistical classifier. Ideally, a feature extractor should only pass the information that is relevant to the particular text classification task (i.e., useful signals). All other information is considered noise and is filtered out. Unfortunately, it is difficult to construct ideal feature extractors for tasks of practical interest. Although it is possible to construct a feature extractor that passes all information losslessly, it may lead to over-training because the classifier learner will attempt to fit the noise as well.

A practical feature extractor is somewhere in between. It lets some but not all relevant information through, and not all the noise is filtered out [Liu and Motoda, 1998]. The amount of relevant information and noise that gets passed to the statistical classifier, i.e.,

the signal-to-noise ratio, sets an upper bound of how well the text classification system can possibly perform, no matter what machine learning algorithm is employed. Therefore, using a good set of features is crucial to the performance of text classification. Moreover, we find in many text classification tasks that it is often more effective to design good features that are well-targeted to the task than varying the classifier.

1.3 Feature Engineering vs. Feature Learning

Despite being an important component of text classification, research on text features often lacks theoretical support and receives much less attention compared to machine learning of classifiers. In many cases, the choice of text features remains highly application-dependent and subjective. For a particular application, features are often picked based on one's understanding to the properties of the application. It is common that there are a few feature choices available, and experimentation is usually the best if not the only way to decide which features should be used. Because it requires engineers to study the task and carry out experiments to try and compare various text features, this process is usually referred to *feature engineering*.

Although satisfying performance for a particular text classification task may be achieved by sufficient investment in feature engineering, it does not easily extend to new tasks, since it requires analysis and understanding of the data and task, and the design and implementation of feature comparison experiments can be time-consuming. Many recent advances in machine learning have greatly improved the accuracy of classifiers. The success of machine learning in classifier design motivates its use in engineering text features.

In this thesis, we are interested in *feature learning*, especially the machine learning technologies that can be applied to text features. There are two goals of feature learning, including learning new features and leveraging relationships among features.

Automatic learning of new features enables more relevant information to be passed to the classifier without adding too much noise, improving the signal-to-noise ratio of text features. Because relevance varies as the application changes, we propose to use supervised learning of new features. Given a set of labeled training data, new, relevant features are selected and added to the feature pool if they satisfy certain criteria that do not depend on

applications.

When we have much more unlabeled data than labeled data, we may observe many features that only appear in unlabeled data. If we use supervised learning, the resulting classifier simply ignores these features, as it does not see any example in the training data. By utilizing feature relationships learned from unlabeled data, we can train a model that is aware of these features. Because both labeled data and feature relationships, which are based on unlabeled data, are used in training, this approach falls into the category of semi-supervised learning.

It should be pointed out that the feature learning proposed in this thesis is not transform-based feature learning. Transform-based feature learning learns a feature transform (c.f. section 2.2.3) from labeled or unlabeled data. The learned transform can be a linear transform or a non-linear transform (e.g., implemented by neural networks). Our proposed feature learning achieves something that feature transforms cannot: we can add new features automatically, and it is easier to control the relative importance of the learned knowledge about features.

1.4 Contributions

The contribution of this thesis is two fold. First, we propose an efficient algorithm to automatically learn new features, i.e., phrase pattern features, from labeled data. Second, we propose semi-supervised feature regularization approaches that explore feature relationships from unlabeled data and use them in training.

Phrase patterns are generalized n-grams with the following additional characteristics: first, gaps between words are allowed in a phrase pattern; second, each slot in a phrase pattern can accommodate more than one token, which could be the word itself or the classes to which the word belongs. The gaps in phrase patterns can overcome the locality of n-grams when non-local behaviors need to be modeled, and the word classes in phrase patterns reduce feature sparsity by enabling one phrase pattern to match many actual word sequences. In this thesis, we propose an algorithm to learn discriminative phrase patterns by recursively growing a phrase pattern tree using a mutual information criterion. Compared to existing approaches, our approach is fully automatic, and is applicable to multi-class

problems.

The second aspect of this thesis concerns a semi-supervised scenario. For many emerging text classification tasks, there are limited amounts of labeled data to train a statistical classifier. Often large volumes of unlabeled data are available but unused in feature extraction. This thesis proposes to leverage relationships among features, which are learned from unlabeled data, to aid the training of classifiers. The goal is not only to better estimate classifier parameters, but to make the classifier aware of the features unseen in labeled data. Specifically, we propose two types of regularization schemes that can be applied to MaxEnt, including feature affinity regularization and feature cluster regularization. We first learn feature affinities and feature clusters from unlabeled data, and then use them to regularize classifier training. Both types of regularizers push the weights of similar features (with high affinity or belonging to the same cluster) closer. There are two effects to the classifier: it prevents the weights of two features from being too different if they are shown to be similar on unlabeled data; and it propagates weights from features seen in training data to unseen features.

To make feature regularizations more practical, we also extend the algorithms to a larger scale, where we have much more unlabeled data to learn feature affinity and clustering from, and high-dimensional n-gram features are used. We experiment with a mixture co-occurrence model that can be learned over multiple machines efficiently. By deriving feature affinity and clustering from this model, we show significantly improved accuracies in Wikipedia alignment move and authority claim classification.

Experiments are carried out for a variety of tasks including movie review sentiment classification, 20 Newsgroups and Reuters-21578 topic classification, talk show speaker role classification, and Wikipedia discussion alignment move and authority claim classification. With significant classification accuracy improvements, we demonstrate the utility of the proposed algorithms over multiple tasks, and we advance the state of the art on talk show and Wikipedia discussion-related tasks.

1.5 Dissertation Overview

In chapter 2, we will review the background work related to this thesis in general. Some work that is related to specific technologies introduced in chapter 3 – 5 will be reviewed in the corresponding chapters for better comparison. The work reviewed in chapter 2 includes commonly used features for text classification, typical feature processing techniques, maximum entropy models (which we will use in this thesis), regularization methods for maximum entropy models, and semi-supervised training of maximum entropy models.

Chapter 3 will introduce our proposed methods for supervised learning of new features. We will study phrase patterns with word and word classes in detail, including efficient algorithms to automatically learn them from labeled data. Several implementation details will be discussed, along with a few practical choices of word classes. Experiments are carried out on speaker role classification for broadcast conversations, Wikipedia discussion alignment move classifications, and Wikipedia discussion authority claim classifications.

Chapter 4 will introduce semi-supervised feature learning methods for text classification, including feature affinity regularization and feature cluster regularization. Fixed point weight propagation will be introduced to improve weight estimation. Different general approaches for learning feature affinity and feature clustering from unlabeled data will be discussed. We will also show that under certain assumptions, both forms of feature regularization are equivalent. The efficacy of proposed approaches will be examined in experiments including movie review sentiment classification, 20 Newsgroups topic classification, and Reuters-21578 topic classification. Word unigram features are used in the experiments, which allows us to focus on performance of different regularization schemes.

In chapter 5, we will investigate ways to scale up the approaches introduced in chapter 4 to the tasks for which large amounts unlabeled data are available and high-dimensional features are common. We will look at a few ways to learn feature affinity and clustering from large unlabeled data set efficiently with the help of parallel computing. Focusing on the Wikipedia discussion-related tasks introduced in chapter 3, we will show how to learn a mixture co-occurrence model and to derive feature affinity and clustering from the entire Wikipedia discussion data. We verify the proposed approaches in experiments including

semi-supervised Wikipedia discussion alignment move and authority claim classification.

Finally, we will conclude this thesis with a detailed summary of contributions in chapter 6, where we will also point out a few possible future directions to extend the work described in this thesis.

Chapter 2

BACKGROUND

In this chapter, we provide some background for the following chapters. First of all, we review the features widely used in various text classification tasks. Several feature processing techniques, including feature selection, feature clustering, and feature transforms will be introduced afterwards. Next, we review the classifier used throughout this thesis, that is, MaxEnt. A few related issues, including regularization and semi-supervised learning of MaxEnt classifiers are also discussed.

2.1 Features for Text Classification

In this section, we review features that are commonly used in text classification, including word n-gram features, phrase pattern features, and linguistic knowledge-driven features.

2.1.1 Word n-gram Features

Word n-grams are tuples of n consecutive words. They have been extensively used in language modeling [Manning and Schuetze, 1999] to predict a word based on its history. An n-gram feature represents the presence (binary feature) or count (integer-valued feature) of an ordered word sequence. In a document or other classification unit, counts are sometimes weighted by word confidence (for speech) or importance (yielding real-valued features).

Single word features, also known as unigram features or bag-of-words (BOW) features, are the simplest n-gram features ($n = 1$). They were originally used with naive Bayes [Duda et al., 2001] in many text classification applications, including topic categorization [Lewis, 1992, Lewis and Ringuette, 1994, McCallum and Nigam, 1998] and sentiment classification [Allison, 2008]. Word features are also widely employed with other statistical classifiers, and sometimes they produce decent performance. Word features were used with SVMs to provide the best movie review polarity classification performance in [Pang et al., 2002], and a similar

setting was used in subsequent work by the same authors [Pang and Lee, 2004, 2005, Thomas et al., 2006]. In this series of work, binary word features were found to be better than integer-valued word features. Weighted word features, e.g., term frequency – inverse document frequency (TF-IDF), are shown to be very effective for topic categorization [Joachims, 1997, 1998], because words related to topics are upweighted and function words are downweighted. They have also been used in classification tasks beyond topic classification. Yu et al. [2007] used TF-IDF in combination with naive Bayes and SVMs to classify ideology for political speech. Martineau and Finin [2009] built a sentiment analysis system using delta TF-IDF features, where term frequency is replaced by the term frequency difference between positive and negative classes.

High-order n-gram features ($n > 1$) have also been used in previous work. They are usually not found to improve over word features in topic categorization; for example, Scott and Matwin [1999] tested the use of n-gram phrases with negative results partly due to over-training. This can be explained by the fact that word features have almost captured all the information for broad topic categories, and additional n-gram features bring no additional gain and can lead to over-training. Nevertheless, several other tasks beyond topic benefit from the use of n-gram features. Barzilay et al. [2000] applied knowledge-driven n-gram signature phrases for broadcast speaker role identification. Liu [2006] used bigrams and trigrams with a MaxEnt classifier for speaker role classification in transcribed broadcast news speech. Kim and Hovy [2006] used unigrams, bigrams, and trigrams as part of the feature set for extracting pros and cons from online reviews. Blitzer et al. [2007] used unigram and bigram features in domain adaptation for sentiment classification. Ifrim et al. [2008] used n-gram features with logistic regression for movie genre, book review genre, and Chinese text topic classification. Gilbert et al. [2009] employed SVMs and TF-IDF-weighted n-grams together with additional features to classify blog comments into agreement and disagreement.

In general, word n-gram features can often achieve satisfying performance despite of their simplicity. Due to the wide usage of word n-gram features in the literature, they will be used as a baseline in this thesis.

2.1.2 Phrase Patterns

Phrase patterns, also known as sequential patterns, loose n-grams, gappy n-grams, and association rules, are defined by a sequence of tokens. Assuming that tokens are words, one can think of sequential phrase patterns as the generalizations of n-grams, by allowing gaps between words. It accounts for certain uninteresting types of variation in natural language (e.g., disfluencies), and it also models long-range dependencies without resorting to high-order n-grams. Phrase pattern features are one emphasis of this thesis as an useful supplement to n-gram features.

The total number of phrase patterns can be found in a text corpus is much larger than that of n-grams. Therefore, we must select the phrase patterns that are the most relevant to our tasks. The automatic selection of phrase patterns is computationally expensive in nature, and there are many sequential pattern mining algorithms [Dong and Pei, 2007] with improved efficiency. Constraints are applied in the pattern mining process to limit the number of selected patterns. Frequency is the most widely used constraint, i.e., only the most frequent patterns in the data are extracted. Efficient algorithms include the *Apriori* method [Agrawal and Srikant, 1994], *prefixSpan* [Pei et al., 2001], *cloSpan* [Yan et al., 2003], etc. Different constraints can be used under certain requirements for different target applications. As an example, Ji et al. [2007] used the frequencies in positive and negative databases as constraints, and discriminative sequential patterns were extracted efficiently. There are also heuristic pattern extraction criteria. For instance, Zhang and Zhu [2007] grew phrase patterns from short patterns with high document frequency and a low χ^2 statistic. The grown patterns are further filtered by a χ^2 threshold. Riloff and Wiebe [2003] and Davidov and Rappoport [2006] first created pattern templates from knowledge. Then the templates are instantiated by the data and qualified patterns are kept.

Due to the costly mining process, phrase patterns are not as popular as n-gram features in text classification systems. They have not been found to improve topic categorization performance. Jaillet et al. [2006] used sequential pattern mining to extract features for Reuters topic categorization. Mixed results were obtained comparing to word features with an SVM classifier. On the other hand, phrase patterns have been successfully applied to

a few tasks beyond topic classification. Wiebe and Riloff [2005] employed automatically extracted phrase patterns to create a subjectivity sentence detector from unannotated text. By learning phrase patterns associated with objectivity, classifiers that achieve substantially higher recall than previous work with comparable precision were built. Sun et al. [2007] used automatically mined phrase patterns to detect erroneous sentences by second language learners, showing that phrase patterns are an effective tool for modeling correct and incorrect sentence structures. Tsur et al. [2010] and Davidov et al. [2010] used automatically extracted phrase patterns with linguistic motivation to recognize sarcastic sentences in Twitter and Amazon product reviews. In an early work, we also used phrase patterns selected using heuristic criteria for unsupervised speaker role classification, and showed that the resulting phrase patterns outperform n-grams on this task [Zhang et al., 2010]. In this thesis, we will propose a general algorithm that learns discriminative phrase pattern features in which word classes can be used flexibly, as determined by the data.

2.1.3 Linguistic Knowledge-driven Features

The knowledge obtained from linguistic study is often helpful for text classification. Part-of-speech (POS) tags can be obtained from automatic POS taggers [Manning and Schuetze, 1999], and they have been used as features in many text classification tasks [Pang et al., 2002, Wiebe and Riloff, 2005, Gilbert et al., 2009, Barbosa and Feng, 2010, Santini, 2004, Feldman et al., 2009]. In addition, structural features extracted by statistical parsers [Manning and Schuetze, 1999] have also been used. However, it is still unclear which structural feature representation is best for general text classification problems, and its application is limited to the languages and genres whose linguistic properties are well studied. Kudo and Matsumoto [2004] examined dependency tree features in review and modality classification, outperforming word features. Gamon [2005] used n-gram, POS n-gram, and constituent structure features for custom feedback classification, while linguistic features alone provided inferior performance. Ng et al. [2006] found that dependency features did not improve review classification. We will explore the use of POS features in the context of phrase patterns. Due to their complexity, dependency features are not studied in this thesis.

2.2 Feature Processing Techniques

In this section, we review a few feature processing techniques for text classification, including feature selection, feature clustering, and feature transforms. These techniques take input feature vectors, process them, and then output new feature vectors with enhanced discrimination, improved robustness, and/or reduced complexity.

2.2.1 Feature Selection

There is usually noise, i.e., information that is irrelevant to the text classification task, embedded in the features. The goal of feature selection is to filter out noisy features, and the output is a reduced-dimensional feature vector. In the case of phrase patterns described above, feature selection is desirable as otherwise the number of phrase patterns is too large to be tractable. Clearly, there is an assumption that the removed features are completely noisy and contain no discriminative information, which may not be true for many applications.

Feature selection is usually supervised, and it is typically accomplished by first ranking features and then discarding the low-ranking ones. Yang and Pedersen [1997] conducted a comparative study of feature ranking methods for topic categorization. The statistics used include document frequency, information gain, and χ^2 statistic. It was found that information gain and χ^2 perform well for topic classification applications. These methods do not take into account redundancies among features. The selected features may be redundant. Peng et al. [2005] proposed minimum-redundancy-maximum-relevance (mRMR) feature selection. The redundancy between feature pairs is computed and included in the mRMR criterion, and the feature selection problem is formulated as an combinatorial optimization problem to find the feature set with the optimal mRMR value. Due to the nature of combinatorial optimization problems, this approach has a high complexity. There are also wrapper approaches [Kohavi and John, 1997] that train a classifier on each set of features and determine the best feature set based on the classification performance on a held-out set. But they tend to be computationally expensive since multiple iterations of classifier training are required.

For some classification frameworks, it is possible to learn a statistical classifier and

select features simultaneously. The most notable approach is to use l_1 regularization in the parameter space [Tibshirani, 1994, Goodman, 2004, Andrew and Gao, 2007]. Because l_1 regularizer favors a sparser solution, many of the parameters learned are zero, effectively disabling the corresponding features. This method requires modification to the training objective function, and therefore may not be applicable to all classifiers.

In this thesis, we will investigate efficient algorithms to perform feature selection for phrase pattern features. Unlike the previously introduced methods where the feature set is given, we must discover and select phrase patterns at the same time, due to the large number of phrase patterns. We will also apply the idea of regularization in this thesis, but the purpose is the propagation of feature weights instead of feature selection.

2.2.2 Feature Clustering

Just as many words have similar meanings, text classification features may be similar to each other. When the similarity between features is sufficiently high, merging them into a single cluster can help remove redundancies. Moreover, noise among features can be removed by collapsing them into a single cluster, if the clustering truly captures the characteristics of the text classification task. The reduced feature dimensionality can alleviate over-training as well, and we get more reliable estimates for feature counts. Feature clustering is in general either unsupervised or hand-specified.

Word classes are easily to obtain, and they have been widely used in text classification. There are multiple ways to derive word classes, including linguistic knowledge-driven (e.g., POS, WordNet synsets), data-driven (e.g., word clustering), and domain knowledge-driven (e.g., hand-specified word classes).

POS tags are coarse word classes, so they are more generalizable features than word features. In addition, POS tags are able to disambiguate different senses of the same word. WordNet [Miller et al., 1990] is a large lexical database of English words and common multi-word expressions are organized as ontology, in which expressions with the same senses are grouped into synsets (concepts), and synsets are organized into trees representing hypernymy/hyponymy. Synsets are word classes in terms of senses, and their hypernyms

provide hierarchical word classes with different level of word senses. Many topic categorization systems have obtained improvements by using WordNet hypernym features [Scott and Matwin, 1998, Fukumoto and Suzuki, 2001, Hotho et al., 2003, Elberrichi, 2006, Amine and Mimoun, 2007]. Relatively few studies (e.g., [Bloehdorn and Hotho, 2004]) have used WordNet in classification beyond topic, due to the high percentage of nouns in WordNet. Another disadvantage of WordNet is the limited language support and coverage.

Data-driven word classes allow the use of in-domain data to be optimized for a particular task. There are supervised and unsupervised methods to cluster words automatically (e.g., [Brown et al., 1992]). They have been applied to topic categorization [Baker and McCallum, 1998, Dhillon et al., 2002, 2003, Bekkerman et al., 2003], resulting in great reduction in feature space with little performance loss. Sometimes gains can be achieved because the word classes generalize better [Baker and McCallum, 1998].

Domain knowledge-driven word classes are often based on keyword lists, where the words in the same list are considered of the same word class. For example, Hillard et al. [2003] and Galley et al. [2004] used the number of positive and negative keywords as part of the feature set for agreement/disagreement classification. Marin et al. [2010, 2011] have also used manually-compiled word classes extensively to classify authority claims in Wikipedia discussions.

Sometimes it is better to use a mixture of words and word classes than using word or word classes only. Frequent words can be used directly for discriminative power, while infrequent words can be substituted by word classes for reliability. Word classes can be used in n-grams and phrase pattern features as well. By choosing a proper mixture of both, a balance between discrimination and reliability can be achieved. In this thesis, we will explore the use of word classes in phrase patterns, and propose algorithms to determine when to use word or word classes automatically.

Word classes are in fact hard feature clusterings. Each feature is assigned a cluster ID, and one feature cannot be in two clusters. Soft feature clusterings assign a vector to each feature, indicating its membership to all clusters. It is more flexible in that each feature can belong to multiple clusters simultaneously, which addresses the polysemy of features. To our best knowledge, no previous work has addressed the problem of utilizing soft feature

clusterings in text classification. We will propose an approach to make use of soft feature clustering in a form of regularization.

2.2.3 Feature Transforms

Once we obtain a feature vector, it can be converted to another feature vector through a linear transform. Many unsupervised methods have been proposed to learn such feature transforms. Principle component analysis [Hastie et al., 2009] is an unsupervised feature transformation, which transforms high-dimensional feature vectors to low-dimensional feature vectors by finding principle directions. However, it is relatively expensive to apply it to text features, since it requires computing a full covariance (or scatter) matrix, and most types of text features are high-dimensional. There are also unsupervised methods to extract latent information from text, which can be considered as new feature vectors. These methods include latent semantic analysis (LSA) [Deerwester et al., 1990, Cai and Hofmann, 2003, Lui et al., 2007], probabilistic latent semantic analysis [Hofmann, 1999], and latent Dirichlet allocation [Blei et al., 2003]. Among them, LSA requires singular value decomposition of large matrices, so it can also be expensive.

Supervised methods enable one to learn feature transforms that are tailored to the task. Linear discriminant analysis [Hastie et al., 2009] finds the feature transform that maximizes the ratio of between-class variance and within-class variance. Recently, due to the popularity of deep learning, some work has been accomplished in transforming text features using deep belief networks [Liu, 2010, Maas and Ng, 2010, Socher et al., 2010]. Deep belief networks are composed of multiple layers of stochastic, latent variables [Hinton and Sejnowski, 1986, Hinton et al., 2006]. These latent variables can be used as a new representation of the original features. The corresponding transforms are no longer necessarily linear.

Usually, data become more separable when features are transformed into a higher-dimensional space. For classifiers that use inner products of features, such as SVMs, the kernel trick is often employed to avoid having to actually compute and store the high-dimensional features [Schölkopf and Smola, 2001]. The direct computation of inner products of high-dimensional features is prevented, and it is instead obtained via a kernel matrix or

kernel function, which defines a similarity between two data samples. Kernel functions are generally specified by hand and selected via supervised evaluation on a development set. Convolution kernels [Haussler, 1999] are proposed for mapping text into a high-dimensional space, and they are common choices of kernel functions for text classification. For example, k-spectrum kernels [Leslie et al., 2002] compare all k-th order n-grams between two text documents. Subsequence kernels [Shawe-Taylor and Cristianini, 2004] and string kernels [Lodhi et al., 2002] compare all subsequences between two documents. Tree kernels [Zelenko et al., 2003] define similarity between two dependency parse trees, by comparing all sub-trees with proper weights. Kernels have been used in various text classification systems, including topic categorization [Jalam and Teytaud, 2001, Zhang et al., 2006, Bloehdorn and Moschitti, 2007, Okanohara and Tsujii, 2009] and review classification [Zhang et al., 2008] with limited success.

We do not focus on feature transforms in this thesis. Rather, the feature regularization scheme can be considered as using the statistics that would be computed for a feature transform as side information to help learn weights for original features. It is better than conventional feature transforms in that we can control the relative impact of the feature transform in a convenient way. It is also more robust to the noise in feature transforms, because original features are always available.

2.3 Maximum Entropy Models

MaxEnt is a type of discriminative classifier widely used in NLP [Berger et al., 1996]. It follows the principle of maximum entropy, which states that, subject to precisely stated prior data (such as a proposition that expresses testable information), the probability distribution which best represents the current state of knowledge is the one with largest information-theoretical entropy [Cover and Thomas, 2006]. Typically, the constraint enforced by labeled training data is that the class-conditional distribution of features computed by the model must be consistent with the empirical distribution of features estimated from data. It is sometimes also called (multi-class) logistic regression or a log-linear model. The decision boundary created by MaxEnt is linear.

For each N-dimensional feature vector $\mathbf{x}$, the MaxEnt model computes the posterior

probability for class $k = 1, 2, \ldots, K$ based on the following formula,

$$p(y = k|\mathbf{x}) = \frac{\exp\left(\sum_{i=1}^{N} \lambda_{ik} x_i + \beta_k\right)}{\sum_{j=1}^{K} \exp\left(\sum_{i=1}^{N} \lambda_{ij} x_i + \beta_k\right)}, \tag{2.1}$$

where λ_{ik}'s are real-valued feature weights which describe how indicative feature i is with respect to class k. They form an $N \times K$ weight matrix $\mathbf{\Lambda}$. The β_k's are the bias terms for each class. They are sometimes called default feature weights if we consider they are the weights for an always fired feature. They represent the model's bias towards each class when no actual feature is fired.

There has been much work published on using MaxEnt in various NLP tasks, many of which involve text classification. Barzilay et al. [2000] employed MaxEnt in radio broadcast speaker role classification, outperforming BoosTexter [Schapire and Singer, 2000]. Similarly, Liu [2006] also found MaxEnt to outperform a hidden Markov model in broadcast news speech speaker role classification. Pang et al. [2002] used MaxEnt for movie review classification, and compared it with an SVM approach. Although the best performance was achieved by SVM, the difference between SVM and MaxEnt was small. With some different choices of features, MaxEnt outperformed SVM. Kim and Hovy [2006] implemented a MaxEnt-based system to identify pro and con reasons in online reviews. Wang et al. [2010] used MaxEnt with words clustered by information bottleneck to classify topics for the Enron email corpus, outperforming the baseline of SVM.

We use the MaxEnt classifier in this work because it has been successful in so many text classification tasks.

2.4 Regularization of Maximum Entropy Models

To learn a MaxEnt model from labeled training data, we find the optimal weights that minimize the objective function composed of negative likelihood and regularizers, where the role of a regularizer is to discourage extreme parameter values, therefore preventing over-training:

$$J(\mathbf{\Lambda}) = -\sum_{(\mathbf{x},y)\in\mathcal{L}} \log p(y|\mathbf{x}) + \sum_{i=1}^{N}\sum_{k=1}^{K} \frac{\lambda_{ik}^2}{2\sigma^2}, \tag{2.2}$$

where the second term on the right is the l_2 regularizer, and σ is the hyper-parameter than controls the strength of this regularization. It can also be considered adding a Gaussian prior with standard deviation σ to the parameters. Because this objective function is convex, gradient methods are used to find the minimum. L-BFGS [Byrd et al., 1995, Zhu et al., 1997] is a commonly used quasi-Newton method for this purpose.

An l_2 regularizer is a simple and effective regularizer. It does not force parameters to go to zero as its gradient near zero approaches zero. Sometimes it is desirable to favor sparse parameters, where some parameters are zero, for performance or computational reasons. It can be achieved by using an l_1 regularizer, and the objective function becomes

$$J_{l_1}(\mathbf{\Lambda}) = -\sum_{(\mathbf{x},y)\in\mathcal{L}} \log p(y|\mathbf{x}) + \sum_{i=1}^{N}\sum_{k=1}^{K} \frac{|\lambda_{ik}|}{\xi}. \tag{2.3}$$

Again, the second term on the right is the l_1 regularizer, and ξ is the hyper-parameter than controls the strength of this regularization. It is equivalent to adding a Laplacian prior to the parameters. The l_1 regularizer is not everywhere differentiable, making gradient methods not directly applicable. Andrew and Gao [2007] proposed an orthant-wise limited-memory quasi-Newton method to efficiently minimize this objective function.

Ng showed in theory [Ng, 2004] that the number of training samples required to train a l_1-regularized MaxEnt grows logarithmically with the number of irrelevant features, whereas that to train a l_2-regularized MaxEnt grows linearly with respect to the number of irrelevant features. Ng also demonstrated this theory in controlled experiments. For real tasks, however, it is unclear whether l_1 or l_2 achieves better performance. There has been little work that compares the performance of both regularizers in practice, and, due to its simplicity, l_2 remains to be the most widely used regularizer for MaxEnt. In our experience, l_2-regularized MaxEnt often outperforms the l_1 counterpart, due to the fact that many complex text classification tasks do rely on a large number of features, although l_1 is able to greatly reduce the model size when storage is a concern.

2.5 Semi-supervised Maximum Entropy Models

Generative classifiers, such as naïve Bayes, compute joint probability distributions for features and labels. In a semi-supervised scenario, the expectation maximization (EM) algo-

rithm can be applied by treating the labels of unlabeled data as missing information [Nigam et al., 2006]. Discriminative classifiers such as MaxEnt only compute the conditional probability of a label given features, and the EM algorithm cannot be directly applied.

Various alternatives for semi-supervised learning have been explored for MaxEnt. Grandvalet et al. proposed entropy minimization in [Grandvalet and Bengio, 2004], which adds an entropy regularization term to the traditional MaxEnt training objective. This regularizer encourages lower entropy of predicted labels on unlabeled data. The intuition behind entropy minimization is the principle used by many semi-supervised learning methods, that is, classification boundaries should lie in low density regions. By favoring lower entropy (thus more certain) predictions on unlabeled data, the decision boundaries are effectively pushed away from high density regions. As with other methods that use this principle, it may not work well if the data has significant overlaps between different classes, where the classifier may get over-confident on the incorrect decisions.

Mann and McCallum [2007] proposed another semi-supervised MaxEnt learning method named expectation regularization (XR). Like entropy minimization, XR also adds a new regularization term to the training objective, specifically the Kullback-Leibler divergence $D(\tilde{p}\|\hat{p})$ between the model-expected label distribution $\hat{p}$ of unlabeled data and a prior label distribution $\tilde{p}$. It prefers models that produce a label distribution closer to the prior, which may be estimated empirically from labeled data or specified by experts. It was shown that XR outperforms entropy minimization on a few tasks [Mann and McCallum, 2007]. Other work has investigated l_2 posterior label distribution regularization for conditional random fields [Subramanya et al., 2010]. XR has also been extended to weakly-supervised MaxEnt learning by utilizing human annotation of salient features, from which a label distribution can be estimated [Mann and McCallum, 2010].

Sandler et al. [2008] proposed regularization with networks of features as a means of semi-supervised training for MaxEnt. A feature network is built based on unlabeled data, and the regularizer pushes the feature weight closer to its neighboring feature weights during training. Our thesis shares the same motivation that unlabeled data is a valuable source for learning relationships among features, but our work focuses on application-independent approach for learning feature relationships, and is thus more general. Furthermore, we

systematically investigate the use of feature clustering in regularization, as well as learning feature clustering for high-dimensional features.

2.6 Summary

In this chapter, we have reviewed commonly used text features, including n-gram features, phrase pattern features, and linguistic knowledge-driven features. We have also reviewed the classifier used throughout this thesis, e.g., maximum entropy models. Conventional regularization methods and semi-supervised learning methods of maximum entropy models have been introduced as well.

Chapter 3

LEARNING PHRASE PATTERNS

Complex features usually span a high-dimensional feature space. Given a fixed number of training samples, estimation of probabilistic models becomes less reliable to estimate probabilistic models when the feature dimensionality gets higher; the classifier can be over-trained and will not generalize well. This is an especially severe problem for tasks where we want to classify complex behaviors with limited labeled data. Using better regularization is one approach to combat over-fitting. Here we explore a method for learning generalizable features as a complementary approach to regularization for improving results in sparse data applications. Specifically, this thesis explores methods for learning phrase patterns with word classes as features for text classification.

Phrase patterns are generalizations of n-grams that allow gaps between words, which make it possible to capture non-local behaviors that cannot be easily captured by n-grams. For example, to capture the characteristic expression of a host in a talk show *"Let's welcome X from Y,"* it takes many n-grams with many different (variable length) instantiations of named entities X and Y. If gaps are allowed, the names can simply be skipped and one phrase pattern will be sufficient, which is especially useful for previously unseen names.

The remainder of the chapter is organized as follows. We first introduce the formal definition of phrase patterns with word classes. Then we describe the algorithm for efficiently learning discriminative phrase patterns. We discuss implementation considerations and also describe the word classes used in the experiments. Three tasks are studied in the experiments and the effect of phrase patterns and different word classes are compared. Finally, we discuss the experimental results and summarize the chapter.

The work introduced in this chapter has been published in [Zhang et al., 2011, 2013], where the author this thesis led and played a major role in the design and implementation of the proposed algorithms. The experiments were carried out by the author.

3.1 Phrase Patterns and Word Classes

Phrase patterns are defined as sequences of words. Unlike n-grams, however, the words are not required to occur consecutively for the phrase pattern to be matched. For example, suppose a phrase pattern P is represented as the word sequence

$$[w_1, w_2, \ldots, w_n].$$

We can represent any sentence S as another word sequence

$$[u_1, u_2, \ldots, u_m],$$

We say that S matches P if there exists a sequence of indices

$$1 \leq i_1 < i_2 < \cdots < i_n \leq m,$$

such that

$$w_j = u_{i_j},\ j = 1, 2, \ldots, n.$$

Namely, the phrase pattern is a subsequence of the sentence when matched. Additional restrictions may apply by, for instance, requiring the length of the gaps be less than a certain size.

Using the concept of sequential patterns or association rules [Agrawal and Srikant, 1994, Wang and Yang, 2005, Dong and Pei, 2007], it becomes natural to incorporate word classes into phrase patterns. We extend the definition of a phrase pattern to include word classes by first defining a *token set* as a word and its associated (context-dependent) classes:

$$t_i = \{u_i, c_i^1, c_i^2, \ldots\ , c_i^C\} = \{t_{i,0}, \ldots, t_{i,C}\}.$$

Here we restrict the choices such that each token set can contain at most one word and any number of word classes. Using the definition above, we can expand the sentence word sequence $[u_1, u_2, \ldots, u_m]$ to a sequence of token sets $[t_1, t_2, \ldots, t_m]$, and a phrase pattern with word classes is defined as a sequence of token sets, e.g.,

$$[s_1, s_2, \ldots, s_n].$$

We say that the sentence matches the phrase pattern if there exists a sequence of indices

$$1 \leq i_1 < i_2 < \cdots < i_n \leq m,$$

such that

$$s_j \subseteq t_{i_j},\ j = 1, 2, \ldots, n,$$

i.e., each token set s_j contains a subset of tokens that may include a word and/or word classes. In other words, the phrase pattern with word classes is a subsequence of the expanded form of the sentence that includes words and/or their context-dependent class labels. The matching of a phrase pattern with classes to a class-annotated sentence is illustrated in Figure 3.1. Note that the use of a word class is specific to the particular phrase pattern it appears in, so the classes learned here are context dependent. Multiple words can be matched using a single word class (e.g., *NEGATIVE_POLARITY* matches both *not* and *hardly*. A combination of words and word classes can match words with specific contexts (e.g., {*right, JJ_POS*} matches *right* only when it is an adjective).

Clearly, phrase patterns with only words are a special case of phrase pattern with word classes. In the following, unless otherwise specified, we will use the term *phrase pattern* more generally to refer to phrase patterns with word classes.

3.2 Learning Discriminative Phrase Patterns

It is computationally expensive to enumerate all the phrase patterns, except on very small text corpora. Even if they were enumerated, the resulting feature vectors would be very high-dimensional, posing challenges to storage and model learning. Therefore, rather than extracting all possible phrase patterns, we propose to learn a set of discriminative phrase patterns prior to model learning.

There has been extensive work in mining frequent sequential patterns, typically with an objective of finding patterns that are contained in more than some predefined number of samples. Many efficient frequent pattern mining algorithms have been proposed, including *PrefixSpan* [Pei et al., 2001] (i.e., prefix-projected sequential pattern mining) and *CloSpan* [Yan et al., 2003] (i.e., closed sequential pattern mining). These algorithms are based

Phrase pattern: ({you}, {NEGATIVE_POLARITY}, {right, JJ_POS})

Sentence 1

Words:	[You]	are	not	[right]	.
POS:	PRP	VBP	RB	[JJ]	
Polarity:			[NEGATIVE]		

Sentence 2

Words:	[You]	were	hardly	[right]	.
POS:	PRP	VBD	RB	[JJ]	
Polarity:			[NEGATIVE]		

Figure 3.1: Example of a phrase pattern matched by two sentences. We expand the words in sentences by word classes (POS and word polarity shown). Boxed elements are matches to the phrase pattern.

on recursive pattern growing, and efficiency is achieved by utilizing the fact that when the frequency of a phrase pattern is below the threshold, there is no need to continue searching for extended patterns as they have even lower frequency. We outline the *PrefixSpan* algorithm applied to frequent phrase pattern mining as algorithm 1. We also define a prefix of phrase pattern $[s_1, s_2, \ldots, s_n]$ as $[s_1, s_2, \ldots, s_l, s'_{l+1}]$ for some $0 < l < n$ and $s'_{l+1} \subseteq s_{l+1}$. Algorithm 1 grows each phrase pattern recursively by appending a new token set or adding a new token to the last token set. The difference between condition (a) and (b) in this algorithm can be explained in the following example. Suppose we are considering different extensions to the prefix $\rho = [s_1, s_2, \ldots, s_n]$. For (a), we look for a new token t such that the new pattern is in the form $\rho' = [s_1, s_2, \ldots, s_n, \{t\}]$. For (b), we look for a new token t such that the new pattern is in the form $\rho'' = [s_1, s_2, \ldots, s_n \cup \{t\}]$.

For efficiency, the projected corpus D' in Algorithm 1 usually does not contain any actual sentences, but rather pointers to the sentences in corpus D and the locations of the latest found token, which will become the starting points for the next scan. Each time an

Algorithm 1: PrefixSpan(D, ρ, f) [Pei et al., 2001]

Input: A corpus D, a prefix pattern ρ, the minimum frequency threshold f

Output: The complete set of phrase patterns P in D with frequency greater than f

Initialize $P \leftarrow \emptyset$;

Scan all the sentences in D once, find a set of tokens A with frequencies of tokens no less than f, such that

(a) the token can be appended to ρ to form a new phrase pattern, or

(b) the token can be added to the last token set of ρ to form a new phrase pattern;

for $a \in A$ **do**

  Create a new phrase pattern ρ' by appending a to ρ;

  $P \leftarrow P \cup \{\rho'\}$;

  Create the ρ'-projected corpus D' from D;

  Call PrefixSpan(D', ρ', f) to obtain a set of phrase patterns B;

  $P \leftarrow P \cup B$;

end

return P

extension is sought after the prefix pattern has grown, the number of sentences that we need to examine reduces, because fewer sentences contain the growing prefix pattern. To deal with that efficiently, a subroutine for projecting the corpus with respect to a prefix pattern is used (algorithm 2) [Dong and Pei, 2007].

Algorithm 2: Create ρ-projected corpus [Dong and Pei, 2007]

Input: A corpus D, a prefix pattern $\rho = [b_1, b_2, \ldots, b_m]$

Output: ρ-projected corpus D'

```
Initialize D' ← ∅;
for s = [a_1, a_2, ..., a_l] ∈ D do
    Find the list of indices 1 ≤ j_1 < j_2 < ··· < j_m ≤ l, where
    b_1 = a_{j_1}, b_2 = a_{j_2}, ..., b_m = a_{j_m} and j_m is as small as possible;
    if success then
        Extract non-empty sub-sentence, in the form of a pointer to the original
        sentence and a new starting location, s' = [a_{j_m+1}, ..., a_l];
        D' ← D' ∪ {s'};
    end
end
return D'
```

Although frequent sequential pattern mining algorithms are very efficient, they are not necessarily optimal for discriminative phrase pattern learning. Not all frequent phrase patterns are discriminative, and many discriminative phrase patterns are not frequent. Indeed, frequent phrase pattern mining is an unsupervised method for feature learning, so it is difficult to optimize for discrimination. If we can add some supervision to the algorithm, we may be able to find phrase patterns that are more relevant to a text classification task. The *ConSGapMiner* algorithm [Ji et al., 2007] extends *PrefixSpan* and addresses the problem of discriminative sequential pattern mining to some extent. Its idea is as follows. Let us assume that there are two corpora, one containing only sentences with positive labels and the other only sentences with negative labels. *ConSGapMiner* mines the phrase patterns with

frequency greater than δ in the positive database and less than α in the negative database, where δ and α are predefined thresholds. A drawback of this algorithm is that it does not easily extend to a multi-class setting. To overcome this, we propose to use an information theoretic criterion for discriminative phrase pattern mining.

3.2.1 The Mutual Information Criterion

Mutual information is the reduction of uncertainty in a random variable after observing another random variable. Mutual information has been shown to be superior to many other feature selection criteria in text categorization tasks [Yang and Pedersen, 1997]. Suppose we have a data set with K classes. Based on the data, we can count the number of occurrences of any binary feature in different classes in the form of a contingency table, which can be used to estimate $p(x, y)$.

The mutual information between feature X and the class variable Y can be computed by

$$I(X;Y) = \sum_{x=0,1} \sum_{y=1}^{K} p(x,y) \log \frac{p(x,y)}{p(x)p(y)},$$

where X is 1 if the associated word or phrase is present and 0 if absent, and the probabilities are estimated from the data using maximum likelihood estimation.[1]

Algorithm 1 uses a frequency threshold as the criterion to select frequent phrase patterns, taking advantage of the prefix-monotonicity of the frequency criterion [Dong and Pei, 2007] to improve the efficiency. Unlike the frequency criterion, the mutual information criterion is not prefix-monotonic. If a sequential pattern satisfies the mutual information criterion, its prefixes do not necessarily satisfy the criterion as well. For example, a phrase pattern may distribute evenly across different classes, and is therefore not discriminative; but appending another word may greatly bias the distribution and make the longer phrase pattern more discriminative. Being unable to terminate the search preemptively based on the prefix pattern, we lose the efficiency of the *PrefixSpan* algorithm. To overcome this, we rewrite

[1] We use unsmoothed empirical frequencies to estimate the probabilities.

the mutual information in the following form,

$$I(X;Y) = \sum_{x,y} p(x|y)p(y) \log \frac{p(x|y)}{\sum_{y'} p(x|y')p(y')}, \tag{3.1}$$

where $p(y)$ is the prior class distribution, which is constant given fully labeled data. It can be shown that, given $p(y)$, $I(X;Y)$ is a convex function of $p(x|y)$ [Liu, 2005]. Let XE be the binary feature indicator of a pattern that extends X with extension E. Suppose we have already collected the statistics in the contingency table for estimating $p(x|y)$ and hence $I(X;Y)$. Since the frequency of an extended pattern is always no greater than that of its prefix,

$$p(XE = 1|y) \leq p(X = 1|y),$$

we can derive an upper-bound for the mutual information of all possible extended patterns

$$\max_{p(XE=1|y) \leq p(X=1|y)} I(XE;Y). \tag{3.2}$$

This quantity can then be used to determine if we should terminate the search for extended patterns: for a phrase pattern, if the upper bound computed by (3.2) is below the threshold θ, then the extended patterns will not be searched.

The maximizer of (3.2) satisfies either $p(XE = 1|Y = i) = 0$ or $p(XE = 1|Y = i) = p(X = 1|Y = i)$ for $i = 1, 2, \ldots, K$, since maximization of a convex function on a compact convex set is always attained on the boundary of the constraints [Rockafellar, 1997]. For a small K, which is the case in most text classification problems, (3.2) can be efficiently calculated by evaluating the boundary points.

3.2.2 Extending PrefixSpan

The above analysis leads to the idea of using two different criteria in the pattern learning process. We can:

1. use the mutual information of the phrase pattern to determine if a phrase pattern is discriminative, and

2. use the upper-bound for mutual information (3.2) against the threshold to determine if any extensions of the phrase pattern may be discriminative.

We refer to the first step as finding *qualified phrase patterns,* and the second step as finding *promising phrase patterns.* Note that they do not imply each other. A qualified phrase pattern may not be promising to search for extensions of, and an unqualified phrase pattern may have discriminative extensions and thus is still promising. To achieve these two steps, we extend *PrefixSpan* into algorithm 3. It is different from algorithm 1 in that, in the scanning of the corpus, it not only finds the qualified patterns, but also keeps track of the promising patterns for further mining. Note that Algorithm 1 is a special case of algorithm 3, in which the same criterion is used to check for qualified and promising patterns. Algorithm 3 can be applied to a wider range of criteria than mutual information. It does not require the criterion to be prefix-monotonic, as long as there is a way to efficiently compute the bounds. Hence, we use the general notation Θ to indicate the criterion for qualified patterns in the algorithm description.

3.2.3 Implementation Considerations and Other Design Decisions

Maximum gap size:

It is counter-intuitive to use unlimited gap size to construct phrase patterns in natural language. Limiting the gap size keeps most of the discriminative phrase patterns, and it can greatly reduce the computational cost in phrase pattern learning. When the algorithm looks for the next qualified and promising token given a prefix pattern, it only needs to check the tokens that are less than the maximum gap size away from the last matched tokens. We use an efficient algorithm proposed in [Ji et al., 2007]. Most of the operations in this algorithm are based on very fast bitset operations. To do that, we need to represent each sentence as a list of bitsets, each bitset corresponds to the location of one particular token in the sentence. For example, given a bitset $[b_1, b_2, \ldots, b_n]$, I is a subset of $\{1, 2, \ldots, n\}$. I indicates the locations where the token appears if $b_i = 1, \forall i \in I$ and $b_i = 0, \forall i \notin I$.

Sentence boundary: There are both intra-sentence and inter-sentence phrase patterns, the former playing more of a syntactic role and the latter conveying discourse information.

Algorithm 3: ExtendedPrefixSpan(D, ρ, Θ)

Input: A corpus D, a prefix pattern ρ, a criterion Θ

Output: The complete set of phrase patterns P in D satisfying criterion Θ

Initialize $P \leftarrow \emptyset$, $A \leftarrow \emptyset$, $B \leftarrow \emptyset$;

Scan all the sentences in D once, find a set of tokens T such that

(a) its element can be appended to ρ to form a new phrase pattern; or

(b) its element can be added to the last token of ρ to form a new phrase pattern;

Assign $x \in T$ to A if it satisfies Θ (qualified tokens)

Assign $x \in T$ to B if its associated bound satisfies Θ (promising tokens)

for $a \in A$ **do**

  Create a new pattern ρ' by appending a to ρ;

  $P \leftarrow P \cup \{\rho'\}$;

end

for $b \in B$ **do**

  Create a new pattern ρ' by appending b to ρ;

  Create the ρ'-projected corpus D' from D;

  Call ExtendedPrefixSpan(D', ρ', Θ) to obtain a set of patterns C;

  $P \leftarrow P \cup C$;

end

return P

Both potentially provide interesting features, but we focus only on the intra-sentence phrase patterns for two main reasons. First, we expect that the discriminative information in our task classification tasks is mostly within sentences, therefore intra-sentence phrase patterns make better features. Second, to use inter-sentence phrase patterns, a large maximum gap size must be used, which adds more computation cost. To limit the phrase patterns within sentences, sentence boundary tokens are placed at the sentence start and end, and token search never goes beyond these tokens. Of course, these tokens may also be useful beyond as a span constraint, if the position of a word in a sentence is a discriminative feature.

Sticky token sets: In the concept of conventional sequential patterns, gaps are allowed between any adjacent token sets. This may not be desirable for text classification or other natural language processing applications. Imagine that we have learned a phrase pattern with three single-word token sets: [{I}, {*agree*}, {*should*}]. If we want to use this phrase pattern to discriminate between agreement and disagreement, we may not want to allow a gap between {I} and {*agree*} in case a negation appears. We call these two token sets sticky, because they are required to appear next to each other. To also learn sticky token sets, we add a condition (c) in algorithm 3 when we search for qualified and promising extension tokens: the token can be appended to ρ without a gap to form a new phrase pattern. The corresponding token has a sticky property set to indicate that its enclosing token set "sticks" with the preceding token set.

Storage: The phrase patterns learned using algorithm 3 naturally form a tree and are stored in this format for efficient search in classification. We start from the root node and recursively check the existence of the child nodes. If a node does not exist in the sentence, we stop the checking for the sub-tree rooted at that node. Every time we go one level down the tree, many nodes are ruled out. Eventually we get to the leaf nodes and the paths to them indicate the phrase patterns existing in the sentence. This procedure is more efficient than checking all phrase patterns linearly, because we do not check patterns for which the prefix does not exist.

We integrate the above consideration into algorithm 3 using an efficient bitset implementation. The major computation of finding qualified and promising extending tokens boils down to checking the presence of each token in the projected sentences, where a projected sentence is a pointer to a sentence with the information of the locations of the last matched token. We describe the procedure of checking the presence of each token in a projected sentence as algorithm 4. Symbols `>>`, `|`, and `&` represent *right shift*, *bitwise or*, and *bitwise and* operation, respectively.

Algorithm 4: FindTokens($s, \{(t_i, b_i)\}_{i=1}^{n}, g$)

Input: A projected sentence s as a bitset representing the locations of the last found token, a list of bitsets b_i representing the locations of each token t_i in the sentence, a maximum gap size g

Output: All tokens T_a, T_b that are present in s within g distance from the locations of the last found token, satisfying condition (a) or (b) in algorithm 3, respectively

Initialize $T_a \leftarrow \emptyset,\ T_b \leftarrow \emptyset$;
$s' = 0$;
for $\eta = 1$ **to** g **do**
    $s' = s' \,|\, (s \gg \eta)$;
end
for $i = 1$ **to** n **do**
    if $s' \,\&\, b_i > 0$ **then**
        $T_a \leftarrow T_a \cup \{(t_i, s' \,\&\, b_i)\}$;
    end
    if $s \,\&\, b_i > 0$ **then**
        $T_b \leftarrow T_b \cup \{(t_i, s \,\&\, b_i)\}$;
    end
end
return T_a, T_b

3.2.4 Choice of Word Classes

In this section, we describe the word classes that are used in our experiments, which include a variety of different text classification tasks in English and Chinese.

Lemma

A lemma is the canonical form of a word, with various inflections removed. The process of obtaining the lemma, i.e., lemmatization, effectively put words into classes. For instance, after lemmatization, the words *go, goes, going, went, gone* are all mapped the word class identified by the lemma *go*. Using the lemma as a word class is useful for inflected languages including English, but not for Chinese. In this thesis, we employ the NLTK WordNet [Miller et al., 1990] lemmatizer for all the tasks involving English data.

Word Shape

Word shapes describe how the letters in words are capitalized; therefore, they only apply to languages with capitalization. By analyzing the capitalization, we can find clues about whether a word is an abbreviation, or is used to represent emotions in online text. Three types of word shape are used: all capitalized, first capitalized, and abbreviation-style capitalization (capital letters and dots interlaced). They apply only to English.

Part-of-Speech

Categorizing words using part-of-speech (POS) tags is a frequently used method to incorporate linguistic knowledge. The same word may be mapped into different categories in different contexts, e.g., *right* may be an adverb, adjective, noun or interjection. In this thesis, we use the Stanford log-linear POS tagger [Toutanova and Manning, 2000, Toutanova et al., 2003]. It contains models for both English and Chinese. We use the tagger to map words into 37 POS categories.

Named Entity

Named entities (NEs) are of particular importance for some text classification tasks. Furthermore, this information is usually difficult to obtain by only looking at words or n-grams. A specialized automatic named entity tagger can use various knowledge sources and produce good NE recognition performance. We use the Stanford conditional random field-based NE recognizer [Finkel et al., 2005] to categorize NEs into four types, including person names, location names, geo-political entities, and miscellaneous names. Only the English NE recognizer is available for this work. Note that the recognizer assigns NEs on a phrase level, and we propagate the NE labels to every word in each detected NE phrase.

LIWC Dictionary

Pennebaker et al. created software that identifies English word classes in text based on a dictionary: Linguistic Inquiry and Word Count (LIWC) [Pennebaker et al., 2001, 2007]. The classes were designed to capture various psychometric statistics, from basic linguistic categories such as articles, to more subjective classes, such as words indicative of causal thinking or emotion words. Using the LIWC dictionary, words can be mapped to 64 classes. The mapping is on a word-by-word basis, and no context information is taken into account.

MPQA Subjectivity Lexicon

To aid sentiment analysis on the Multi-perspective Question Answering (MPQA) Opinion Corpus, Wilson et al. [2005] compiled a prior-polarity subjectivity lexicon. Words that are subjective in most contexts were marked strongly subjective, and those that may only have certain subjective usages were marked weakly subjective. In the mean time, for each subjective word, a prior polarity is provided, indicating whether the word expresses positive or negative polarity out of context. Based on this, we use the lexicon to map words into four overlapping classes: strongly subjective, weakly subjective, positive, and negative. To identify the class for a word, the lexicon requires that the text is lemmatized and POS-tagged.

Manual Word Classes

The word classes introduced above are general purpose. They can be used for many tasks, but may not be optimal for a specific task. To create domain-specific word classes, we use various word lists constructed by linguists who have looked at data related to some of our tasks. Each word list corresponds to a word class, and all words are in full form.

Automatic Word Classes

It is possible to induce domain-specific word classes automatically without human annotation. They can be learned from unlabeled text data using word clustering algorithms, such as the one introduced by Brown et al. [1992]. This algorithm was designed to estimate class-based language models, and the word clustering objective function is the log likelihood of the training data computed using a first-order Markov model. In our experiments, we use the implementation of the algorithm in SRILM [Stolcke, 2002] to generate data-driven word classes derived for each corpus. The number of clusters (classes) is varied among 10, 100, and 1000. The count of the most frequent words that are kept out of any cluster is also varied among 0, 10, 100, and 1000. The optimal choice of these values is determined on the development set.

3.3 Experiments

We select a few text classification tasks to evaluate the effect of phrase patterns. We do not use topic classification tasks, as many studies have shown that n-gram or even word features can work very well, and we do not expect phrase patterns to capture more topic information than n-grams. Instead, we select three high-level classification tasks which are orthogonal to the topic information of the document. These tasks are speaker role recognition, alignment move classification, and authority claim classification. All three tasks use data from spoken or online written interactive conversations, and we carry out experiments on both English and Chinese data. For the Chinese data, word segmentation is always performed.

In all experiments, we first learn phrase patterns on the labeled training data. For comparison, a baseline classifier is trained with only n-gram features. Once a set of task-

dependent phrase patterns is obtained, n-gram and phrase pattern features are extracted for both training and test data. For a fair comparison, we use trigrams and phrase patterns of length up to three. A MaxEnt classifier is then trained and tested on each feature set.

The experiments listed in this section are all carried out in the form of five-fold cross validation. The training, development, and evaluation data are randomly sampled from the entire data set, and we maintain their ratio as 60%:20%:20%. The hyper-parameters (mutual information threshold, maximum gap size) for phrase pattern learning are tuned on the development set.

3.3.1 Speaker Role Recognition

In talk shows or broadcast conversations, participants are in various institutional roles. Hosts run the shows and usually control the agenda, while invited guests introduce their experience or opinions with respect to some topics. Automatic recognition of speaker roles captures this information. Initial work on speaker role recognition [Barzilay et al., 2000] was on broadcast news, categorizing speakers into three categories: anchor, journalist, and guest. The authors employed a large number of lexical and structural features, and the feature weights were learned on labeled training data via Boostexter or MaxEnt. An accuracy of 80% was achieved on the ASR-derived transcripts. Liu [2006] studied the classification of speaker roles on TDT-4 Mandarin broadcast news audio data. Hidden Markov and MaxEnt models were used to label the sequence of speaker turns with the roles anchor, reporter, and other, based on n-gram features extracted from human transcripts. The algorithm reached 80% classification accuracy. Vinciarelli [2007] proposed to combine social network and word cues on speaker clustered news bulletins, and achieved an accuracy of 85% on a task of classifying six speaker roles: anchor, secondary anchor, guest, interview participant, abstract (speaker who gives summary at the start of the show), and meteo (speaker who gives a weather report). We have also tried previously using both structural and lexical features for this task using unsupervised clustering [Hutchinson et al., 2010]. Structural features reflect the temporal behaviors of speech, while lexical features focus on the speech content.

Phrase pattern features [Zhang et al., 2010] have been shown to be useful for unsupervised speaker role classification, where speaker roles include host, expert guest, and soundbite. Here, we examine the effect of phrase pattern features compared to n-gram features in a supervised setting. We use lexical features only for more easily contrasting experiments involving different lexical features.

Five speaker roles are defined: host, guest participant, audience participant, reporting participant, and a blanket category "other." Invited speakers are considered guest participants, and journalists are labeled as reporting participants. 48 English and 90 Mandarin talk show recordings are annotated for speaker roles using human transcripts. The same data set was also used in [Zhang et al., 2011]. A subset was annotated by multiple annotators to evaluate annotation consistency. The inter-annotator agreement of speaker role annotation is $\kappa = 0.67$ for English and $\kappa = 0.78$ for Mandarin. For the recordings in that subset, the inconsistent annotations were reconciled by annotators, and the reconciled version is used as the ground truth.

We compare the performance of different lexical features on reference human transcripts and automatic speech recognition (ASR) output, using the SRI Decipher ASR system [Hwang et al., 2007, Lei et al., 2009]. Due to the highly conversational nature of the corpora, ASR error rate on these data sets is high. The word error rate on the English evaluation data is 22.8%, and the character error rate on the Mandarin Chinese evaluation data is 38.6%. When the ASR output is used, we extract the n-gram features, learn the phrase pattern features, and train the classifier without any human transcripts. Thus we can evaluate how these features adapt to the automatic but erroneous ASR transcripts. For both reference and ASR conditions, we use the speaker turn boundaries from the human transcripts and evaluate the turn-level speaker role accuracy.

In our preliminary experiments, we found that using a MaxEnt classifier with a global constraint is more effective for this task than a sequential model. The global constraint enforces that turns from the same speaker have the same role label. In other words, suppose that $t_i,\ i = 1, 2, \ldots, n$ are all the turns of a particular speaker, and $p(r|t_i)$ is the posterior probability of role r for turn t_i predicted by the MaxEnt model; then the final speaker roles

for these turns are

$$\hat{r} = \underset{r}{\operatorname{argmax}} \prod_{i=1}^{n} p(r|t_i).$$

3.3.2 Alignment Move Detection

In social media such as online discussion forums, participants often exchange their opinions on various topics. Arguments and debates can occur when different participants hold different and even opposite opinions, especially on controversial topics. Alignment moves in discussions are defined as participants' attempts to positively or negatively align themselves to other participants by showing agreement and dissent, respectively [Bender et al., 2011, Morgan et al., 2013]. The detection of alignment moves creates a graph of interaction about who supports whom and who expresses dissent toward whom, which is potentially useful in analyzing sub-group dynamics in discussions.

In this work, we use Wikipedia talk page data for alignment move detection. Wikipedia talk pages are discussion pages open to the world about Wikipedia articles. Each talk page is associated with a Wikipedia article, with the talk page usually discussing improvements to the corresponding Wikipedia article. There may be multiple disjoint discussions in a talk page, each covering a specific problem with or section of the article. The Wikipedia talk page data covers a wide variety of topics, but it is relatively constrained in terms of types of interactions. However, the language used in talk pages is quite diverse due to the nature of online text.

Detailed descriptions of the data used in this task can be found in [Bender et al., 2011, Morgan et al., 2013]. 211 English Wikipedia talk page discussions and 225 Chinese Wikipedia talk pages were collected and annotated. The annotation task involves indicating, for each post in these discussions, whether there is a positive alignment move and/or a negative alignment move, highlighting the associated sentences. A positive alignment may contain cues such as explicit agreement, rephrase, thanking, and positive reference to a previous speaker's point. A negative alignment may contain cues such as explicit disagreement, doubting, sarcasm, criticism, insult, and dismissing. Many of the cues are rather implicit and hard to code. The inter-annotator agreement is $\kappa = 0.50$ for English, and $\kappa = 0.64$ for

Chinese, indicating that this is a difficult task even for humans. Another challenge is that alignment moves are sparse throughout the data. Less than a quarter of the posts contain alignment moves, which does not give a lot of training examples, especially in light of the diverse language that can be used to express alignment moves. We build two sentence-level maximum entropy classifiers, one for positive alignment vs. none, and another for negative alignment vs. none. The reason for using two classifiers instead of one is because a complex sentence may be labeled as both positive alignment and negative alignment. As the discussions are labeled by multiple annotators, we use a reconciliation procedure described in [Bender et al., 2011]. Basically, a sentence is considered as having alignment move if at least one annotator annotated it as an alignment move. The detection performance is evaluated using sentence-level F-score. Manual word classes are compiled by annotators. For the English Wikipedia-related tasks, we use 38 overlapping word lists including agreement keywords, disagreement keywords, swear words, and so on. For the Chinese Wikipedia-related tasks, we use 19 overlapping word lists.

3.3.3 Authority Claim Detection

Authority claim detection is another task associated with Wikipedia talk pages. To establish credibility in discussions, Wikipedia participants often attempt through varied means to demonstrate their knowledge or experience, which is used to support their opinions. An authority claim is an attempt a participant makes to claim herself credible. It is usually achieved by showing her knowledge or experience with respect to a topic, or using some external evidence to support herself [Morgan and Oxley, 2010, Oxley et al., 2010]. The most frequent types of authority claims are external claims, where external sources are cited, and forum claims, where participants show that they know the forum policies and norms of discussion by referring to specific Wikipedia rule and policy pages. Marin et al. [2010] studied the problem of authority claim detection in Wikipedia talk pages. Unigram-based features and sentence complexity features produced similar results. The incorporation of interaction words, which are extracted by comparing the corresponding Wikipedia talk pages and main pages, increased feature robustness. Marin et al. [2011] further investigated the

detection of forum authority claims with various syntactically-motivated features, showing that they can provide complementary information to that captured by n-grams.

The data for authority claim detection is also introduced in [Bender et al., 2011, Morgan et al., 2013]. 339 English Wikipedia talk page discussions and 225 Chinese Wikipedia talk page discussions were collected. Each post in the discussion is labeled with the presence or absence of an external claim and/or a forum claim. The inter-agreement agreement is $\kappa = 0.59$ for English and $\kappa = 0.72$ for Chinese. The reconciliation of annotations is done similarly as for alignment moves. Authority claims also have a data sparsity problem, with only about 20% of the posts containing authority claims. We build two maximum entropy classifiers for external and forum claims, respectively. The detection performance is evaluated using sentence-level F-score.

3.3.4 Results

Experimental results are reported in tables 3.1-3.7, using average accuracies for cross validation for speaker role recognition experiments, and average F-scores for alignment move and authority claim classification experiments. The best results obtained for each task are indicated with bold font. The results with one star (*) are significantly different with $0.01 < p \leq 0.05$ from the counterpart or the baseline. The results with two stars (**) are significantly different with $p \leq 0.01$ from the counterpart or the baseline. The significance levels of accuracies are computed by paired t-test, and those of F-scores are estimated by the approximate randomization test [Yeh, 2000]. Note that in some cases a smaller average difference can be more significant because the benefit is more consistent in the paired comparison.

In table 3.1, we first present the comparison between the original *PrefixSpan* and the extended *PrefixSpan* in the context of text classification performance with no word classes in phrase patterns. Out of the twelve experiments, the extended *PrefixSpan* significantly outperforms the original *PrefixSpan* in eight experiments in terms of classification performance, and is only significantly worse in one experiment.

Next, we present the experimental results for each task. We show in detail how different

Task	*PrefixSpan*	extended *PrefixSpan*
English speaker role (Ref.)	**87.1%****	86.9%
English speaker role (ASR)	85.5%	**85.6%****
Chinese speaker role (Ref.)	85.6%	**85.8%***
Chinese speaker role (ASR)	76.9%	**77.8%****
English positive alignment	36.4%	**40.5%****
English negative alignment	37.0%	**38.9%****
Chinese positive alignment	29.0%	**31.2%****
Chinese negative alignment	30.5%	**31.2%****
English external authority	46.7%	**47.7%**
English forum authority	43.4%	**46.0%****
Chinese external authority	29.0%	**31.8%**
Chinese forum authority	**35.7%**	33.5%

Table 3.1: Comparison between the original and extended *PrefixSpan* in text classification. The numbers in the first four rows are accuracies, and the other numbers are F-scores.

word classes impact the classification performance when using phrase pattern features. In these experiments, we use systems with only n-gram features as baselines. Then we add phrase pattern features with different types of word classes (section 3.2.4) for comparison. Word classes are available only in cases where phrase patterns are used.

Pattern?	Word class	Ref.	ASR
no	N.A.	85.8%	85.0%
yes	none	86.9%	85.6%
yes	lemma	86.8%	85.4%
yes	POS	86.2%	85.8%
yes	NE	86.9%	84.7%
yes	LIWC	86.0%	**85.9%**
yes	MPQA	86.5%	**85.9%**
yes	automatic	**87.1%***	85.6%

Table 3.2: English speaker role recognition results (accuracy) on reference (Ref.) and ASR transcripts with different word classes

Pattern?	Word class	Ref.	ASR
no	N.A.	84.6%	70.2%
yes	none	**85.8%***	**77.8%****
yes	POS	84.8%	74.5%**
yes	automatic	85.7%**	77.2%**

Table 3.3: Chinese speaker role recognition results (accuracy) on reference (Ref.) and ASR transcripts with different word classes

Surveying all the experimental results in tables 3.2-3.7, we see eleven out of twelve series of experiments with improvements by adding phrase pattern features, among which six

Pattern?	Word class	Positive	Negative
no	N.A.	38.1%	38.8%
yes	none	40.5%	38.9%
yes	lemma	40.2%	38.8%
yes	word shape	40.0%	39.3%
yes	POS	39.0%	38.6%
yes	NE	40.5%	38.9%
yes	LIWC	39.0%	38.7%
yes	MPQA	39.2%	**40.5%***
yes	manual	**40.8%**	39.4%
yes	automatic	40.7%	**40.5%***

Table 3.4: English alignment move results (F-score) with different word classes for positive and negative alignment moves

Pattern?	Word class	Positive	Negative
no	N.A.	26.7%	29.7%
yes	none	31.2%	31.2%
yes	POS	32.7%*	30.7%
yes	manual	**33.9%****	**31.5%**
yes	automatic	30.9%*	30.6%

Table 3.5: Chinese alignment move results (F-score) with different word classes for positive and negative alignment moves

Pattern?	Word class	External	Forum
no	N.A.	**49.5%**	46.5%
yes	none	47.7%	46.0%
yes	lemma	48.0%	46.7%
yes	word shape	48.2%	45.8%
yes	POS	48.6%	45.1%
yes	NE	47.7%	46.0%
yes	LIWC	48.9%	46.5%
yes	MPQA	47.8%	45.5%
yes	manual	48.0%	**46.8%**
yes	automatic	48.9%	46.1%

Table 3.6: English authority claim results (F-score) with different word classes for external and forum authority claims

Pattern?	Word class	External	Forum
no	N.A.	32.2%	32.3%
yes	none	31.8%	33.5%
yes	POS	**34.3%**	**40.3%***
yes	manual	30.3%	37.9%
yes	automatic	31.4%	35.6%

Table 3.7: Chinese authority claim results (F-score) with different word classes for external and forum authority claims

improvements are significant (all speaker role cases except for English reference transcripts, negative alignment moves in English, positive alignment moves in Mandarin, and forum authority claims). The only case where we did not see improvement is the case of English external authority claims (table 3.6), for which URLs are found to be highly informative. This shows that, in general, adding phrase patterns can do better than n-grams features alone for most tasks.

In the speaker role recognition experiments (table 3.2 and 3.3), phrase patterns improve the performance not only when using reference transcripts, but also with ASR transcripts. The results show the robustness of phrase patterns given erroneous ASR output. Furthermore, we observe a greater improvement for Chinese ASR speaker role recognition than for English ASR speaker role recognition. It can be explained by the fact that the error rate of Chinese Mandarin ASR is higher, causing a lower baseline when training and testing on ASR transcripts. Phrase patterns are more robust to ASR errors, hence they are able to achieve big improvements when employed.

We also observe that phrase patterns with word classes usually work better than without word classes, although we do see the reverse in Chinese speaker role recognition (table 3.3), which may be partly due to the fact that we do not have a lot of word class options. Since the tasks are very different, we do not expect one type of word class to be best for all cases. The goal of these experiments is to show that the performance of phrase patterns can be improved with properly chosen word classes. Overall, automatic word classes offer the most consistent improvements across different tasks (five out of twelve experiments with significant improvements in table 3.2-3.7). That is of practical value because little human effort is needed to use this type of word class. Moreover, compared to POS, automatic word classes perform better on average (POS leads to three out of twelve significant improvements). Automatic word classes are learned on a task-by-task basis, which allows the word classes to fit the domain, unlike general word classes, which do not capture much task-dependent information. Manual word classes that are specifically constructed for the tasks often achieve the top performance (four out of eight experiments in table 3.4-3.7). Hence, if domain experts are available, designing task-specific word classes and incorporating them using phrase patterns is a good approach. This method also provides alternative supervi-

sion to data annotation, especially when the data to be annotated is limited. Off-the-shelf word classes, such as MPQA, would be expected to help with alignment moves more than authority claims (table 3.4 vs. 3.6). However, due to the complexity of alignment moves, they are the best performer only in one of the two alignment move experiments (table 3.4). One explanation for this is that it is not uncommon for an alignment move to contain words with opposite sentiment.

The actual number of phrase patterns extracted varies with the applications and data; typically it is on the order of thousands to tens of thousands. In table 3.8, we show some example patterns extracted during the English Wikipedia alignment experiments. The upper-cased words represent word classes. Classes that were manually designed specifically for this task were frequently used, including AGREEMENT (e.g., *right*, *agree*, *true*), DISAGREEMENT (e.g., *doubt*, *inappropriate*), ALIGNMENT (the union of the AGREEMENT and DISAGREEMENT word lists), MODAL (e.g., *could*, *should*), and NEGATIVE_DISCOURSE_MARKER (e.g., *however*, *but*, *nevertheless*). MPQA_STRONGSUBJ, MPQA_WEAKSUBJ, and MPQA_POSITIVE are word classes defined in MPQA subjectivity lexicon that indicate strong subjectivity, weak subjectivity, and positive sentiment, respectively. The hash sign (#) prefix indicates a sticky token which must immediately follow the preceding token. These phrase patterns provide more flexible matching capability than n-grams do, while still capturing the desired discriminative information for the target text classification task.

3.4 Summary

We have presented phrase patterns as useful features for text classification beyond topic; we have also introduced an efficient way of learning discriminative phrase pattern features automatically. Mutual information is used as the criterion to search for phrase patterns, and the upper-bound of the mutual information is used to terminate the search early. The computation of the upper-bound is simple, requiring only the statistics of the prefix pattern. We have tested the phrase pattern learning algorithm on three text classification tasks, outperforming baselines using n-gram features. The introduction of context-dependent word classes further improves the performance in all but one case, with the most consistent benefit

i AGREEMENT MODAL
i ALIGNMENT you
i #ALIGNMENT you
a POSITIVE #idea
ALIGNMENT with WIKIMARKUP
NEGATIVE_DISCOURSE_MARKER i ALIGNMENT
sorry but DISAGREEMENT
i think ALIGNMENT
not MPQA_STRONGSUBJ you
MPQA_WEAKSUBJ i MPQA_POSITIVE

Table 3.8: Phrase pattern examples

coming from automatically learned classes.

Chapter 4

LEVERAGING FEATURE RELATIONSHIPS LEARNED FROM UNLABELED DATA

Semi-supervised learning techniques are used in many problems to leverage unlabeled data [Zhu, 2005]. Applying semi-supervised learning to text classification is desirable due to limited labeled text and increasing amount of unlabeled text. Suppose we are given a set of label data $\mathcal{L}$ and unlabeled data $\mathcal{U}$ that come from the same distribution, and the goal is to learn a classifier based on both $\mathcal{L}$ and $\mathcal{U}$.

In this chapter, we attempt to use unlabeled data in semi-supervised learning in a different way. Instead of regularizing the labels or posteriors on the unlabeled data, we regularize the feature weights based on the structure of features learned from unlabeled data. This approach is motivated by the fact that the feature space in text classification problems is typically very high-dimensional, so many features are not observed in a limited labeled training data set. Since the posterior predictions are weak if they are only based on a limited feature set, it may be more productive to focus on leveraging information about the features in the unlabeled data. Specifically, we propose feature affinity regularization (FAR) and feature cluster regularization (FCR), which learn feature affinity and clustering information from unlabeled data, and then incorporate it in the training objective in forms of regularization. As a result, we favor models which assign similar weights to similar features. With FAR/FCR, even if we only observe a small number of features on labeled data, the model will still be aware of other unseen features that are similar to the observed features. Therefore, we can train a model that generalizes better to new data. The method applies to both inductive and transductive classifiers, depending on whether the test data can be incorporated in the unlabeled set when learning feature correlation.

Similar forms of regularization are proposed as regularization with networks of features, in work by Sandler et al. [2008], also motivated by the challenges of high-dimensional features

in text classification. They show that the Gaussian form of their regularization is equivalent to the locally linear embedding (LLE) method for transforming high-dimensional data [Roweis and Saul, 2000] without the dimensionality reduction, and that their regularization achieves better results than the standard reduced-dimension LLE for two text classifications. The key differences in our work are the use of l_2 on seen feature weights only, the introduction of fixed point weight propagation (which improves convergence), and the mechanism for representing feature similarity. The key differences in our work are the mechanism for representing feature similarity. In [Sandler et al., 2008], the feature similarity measure is application-dependent and requires external knowledge. In contrast, our work represents feature affinity and cluster purely based on the unlabeled data, which is computationally cheap to obtain and can generalize reasonably well across tasks. Our investigation of FCR with different clustering techniques is also beyond the focus of their paper.

Another method that explores feature similarity (or more precisely, feature correspondence) from unlabeled data is SCL [Blitzer et al., 2006]. SCL can be applied to domain transfer, where source and target domains are unmatched, and labeled data is only available in the source domain. SCL defines a set of pivot features which are common across domains, and constructs linear predictors to predict those pivot features from unlabeled data in both domains. A feature in the source domain is said to correspond to a feature in the target domain if they are both useful in predicting the pivot features, and a linear combination of them is used as a new feature in supervised learning. FAR/FCR is similar in motivation, but is different from SCL in the following respects: it does not require hand-picking of pivot features, and it is designed for within-domain semi-supervised learning (vs. domain transfer).

In the following sections, we will formally define feature affinity regularization and feature cluster regularization in the context of MaxEnt classifiers. Their convexity will be shown. We will also discuss a few extreme cases of feature affinity regularization, and show that both forms of feature regularization are in fact equivalent under certain assumptions. Fixed point weight propagation will be proposed as a measure to improve the update of parameters. We will also list some methods to learn feature affinity and clustering from unlabeled data. Finally, these approaches are evaluated in experiments including movie review

sentiment classification, Wikipedia alignment move classification, and Wikipedia authority claim classification.

Part of the work introduced in this chapter has been published in [Zhang and Ostendorf, 2012].

4.1 Feature Affinity Regularization

We introduce some additional notations for the discussions in the following sections. Once we include the unlabeled data with the labeled training data, the number of features modeled by MaxEnt will be greater than the case where only labeled training data is used, although, using (2.2), only the weights of features observed in labeled training data are non-zero. Suppose the number of features that appear in labeled training data, i.e., *seen features*, is N_s, and the number of features that appear only in unlabeled data, i.e., *unseen features*, is N_u. The total number of features is thus $N = N_s + N_u$. We index the features in such a way that features $1, 2, \ldots, N_s$ are all seen features, and features $N_s+1, N_s+2, \ldots, N$ are all unseen features. Note that we do not include the default feature in the feature set, although in some implementations, it is considered as a feature whose value is always one.

4.1.1 Definition

The FAR regularizer is defined as

$$J_{\text{FAR}}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j,\ i<j} w_{ij}(\lambda_{ik} - \lambda_{jk})^2, \tag{4.1}$$

where w_{ij}'s are elements of a symmetric matrix $\mathbf{W}$. We call $\mathbf{W}$ a *feature affinity matrix* as it encodes affinity information between pairs of features. The larger w_{ij} is, the more similar feature i and feature j are. All w_{ij}'s are non-negative, so $J_{\text{FAR}}(\mathbf{\Lambda}) \geq 0$.

To use FAR in MaxEnt model learning, we add J_{FAR} to the training objective function (2.2), resulting in a new objective function,

$$J(\mathbf{\Lambda}) = -\sum_{(\mathbf{x},y)\in\mathcal{L}} \log p(y|\mathbf{x}) + \sum_{i=1}^{N_s}\sum_{k=1}^{K}\frac{\lambda_{ik}^2}{2\sigma^2} + \gamma J_{\text{FAR}}(\mathbf{\Lambda}). \tag{4.2}$$

The first two terms are negative loglikelihood and an l_2 regularizer as used in the conventional MaxEnt, respectively. Note that the summation in l_2 only includes all the seen features. Unseen features are not subject to l_2 regularization, and the reason will be clear shortly. $\gamma > 0$ is the hyper-parameter that controls the strength of FAR.

It can be shown that $J_{\text{FAR}}(\mathbf{\Lambda})$ is convex with respect to $\mathbf{\Lambda}$.

Proof. To prove its convexity, we only need to show that, given $N \times K$ real matrices $\mathbf{X}$ and $\mathbf{Y}$, for any $0 \leq \alpha \leq 1$, the following inequality holds.

$$J_{\text{FAR}}(\alpha\mathbf{X} + (1-\alpha)\mathbf{Y}) \leq \alpha J_{\text{FAR}}(\mathbf{X}) + (1-\alpha) J_{\text{FAR}}(\mathbf{Y}). \tag{4.3}$$

Each side of this inequality can be computed as the following.

$$J_{\text{FAR}}(\alpha\mathbf{X} + (1-\alpha)\mathbf{Y}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}\left[\alpha x_{ik} + (1-\alpha)y_{ik} - \alpha x_{jk} - (1-\alpha)y_{jk}\right]^2 \tag{4.4}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}\left[\alpha(x_{ik} - x_{jk}) + (1-\alpha)(y_{ik} - y_{jk})\right]^2, \tag{4.5}$$

$$\alpha J_{\text{FAR}}(\mathbf{X}) + (1-\alpha) J_{\text{FAR}}(\mathbf{Y}) = \frac{\alpha}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}(x_{ik} - x_{jk})^2 + \frac{1-\alpha}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}(y_{ik} - y_{jk})^2 \tag{4.6}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}[\alpha(x_{ik} - x_{jk})^2 + (1-\alpha)(y_{ik} - y_{jk})^2]. \tag{4.7}$$

The difference between two terms is

$$J_{\text{FAR}}(\alpha\mathbf{X} + (1-\alpha)\mathbf{Y}) - [\alpha J_{\text{FAR}}(\mathbf{X}) + (1-\alpha)J_{\text{FAR}}(\mathbf{Y})] \tag{4.8}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}\left([\alpha(x_{ik}-x_{jk}) + (1-\alpha)(y_{ik}-y_{jk})]^2 - \alpha(x_{ik}-x_{jk})^2 - (1-\alpha)(y_{ik}-y_{jk})^2\right) \tag{4.9}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}\left[\alpha^2(x_{ik}-x_{jk})^2 + (1-\alpha)^2(y_{ik}-y_{jk})^2 + 2\alpha(1-\alpha)(x_{ik}-x_{jk})(y_{ik}-y_{jk})\right.$$
$$\left.-\alpha(x_{ik}-x_{jk})^2 - (1-\alpha)(y_{ik}-y_{jk})^2\right] \tag{4.10}$$

$$= -\frac{\alpha(1-\alpha)}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}\left[(x_{ik}-x_{jk})^2 + (y_{ik}-y_{jk})^2 - 2(x_{ik}-x_{jk})(y_{ik}-y_{jk})\right] \tag{4.11}$$

$$= -\frac{\alpha(1-\alpha)}{2}\sum_{k=1}^{K}\sum_{i,j\ i<j} w_{ij}\left[(x_{ik}-x_{jk}) - (y_{ik}-y_{jk})\right]^2 \tag{4.12}$$

$$= -\frac{\alpha(1-\alpha)}{2}J_{\text{FAR}}(\mathbf{X}-\mathbf{Y}) \tag{4.13}$$

$$\leq 0. \tag{4.14}$$

□

Therefore, overall $J(\mathbf{\Lambda})$ is still convex with respect to $\mathbf{\Lambda}$, and we can use L-BFGS to minimize J without the risk of being stuck at local minima. The gradient of $J_{\text{FAR}}(\mathbf{\Lambda})$ can be computed as the following. In this derivation, δ_{ij} is the Kronecker delta.

$$\frac{\partial J_{\text{FAR}}(\mathbf{\Lambda})}{\partial \lambda_{ik}} = \sum_{n,j,\ n<j} w_{nj}(\lambda_{nk}-\lambda_{jk})\frac{\partial(\lambda_{nk}-\lambda_{jk})}{\partial\lambda_{ik}} \tag{4.15}$$

$$= \sum_{n,j,\ n<j} w_{nj}(\lambda_{nk}-\lambda_{nk})(\delta_{ni}-\delta_{ji}) \tag{4.16}$$

$$= \sum_{j,\ i<j} w_{ij}(\lambda_{ik}-\lambda_{jk}) - \sum_{n,\ n<i} w_{ni}(\lambda_{nk}-\lambda_{ik}) \tag{4.17}$$

$$= \sum_{j} w_{ij}(\lambda_{ik}-\lambda_{jk}). \tag{4.18}$$

Intuitively, the gradient pushes λ_{ik} towards λ_{jk} if $w_{ij} > 0$, therefore similar features get similar weights.

4.1.2 Analysis

The effect of FAR is two-fold:

1. It performs regularization on the seen feature weights to prevent over-training. This is similar to other conventional regularizers such as l_1 and l_2, but with the participation of unseen features.

2. It propagates weights from seen features to unseen features. This improves the model's capability to handle unseen features, and it cannot be accomplished by conventional regularizers.

To better understand both effects, let us look at the following examples. Imagine that there are two seen features A and B and an unseen feature C. Further assume that the

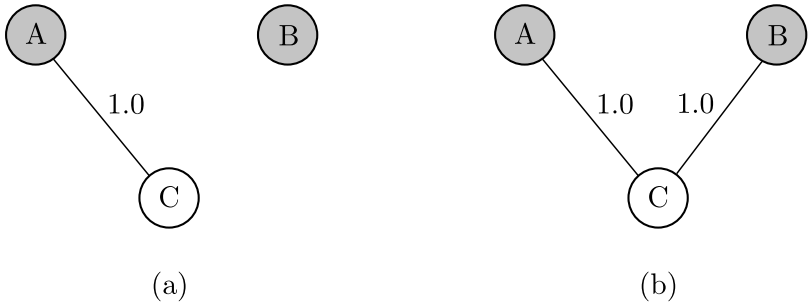


Figure 4.1: Example feature graphs: (a) an unseen feature is connected to only one seen feature, (b) an unseen feature is connected to two seen features

classification problem is two-class. Consequently, the feature weights for positive and negative classes are opposite, and we only need to look at the feature weights for positive class, denoted as $\lambda_i = \lambda_{i1}$. If the C is only connected to A (figure 4.1(a)), i.e., only A and C are similar features, then, intuitively, we should set $\lambda_C = \lambda_A$. This can be accomplished exactly by FAR, as the only term in the objective function involving λ_C is $\frac{\gamma}{2}(\lambda_A - \lambda_C)^2$, whose minimizer is $\lambda_C = \lambda_A$.

The above behavior cannot be achieved, however, if we include l_2 regularization on both seen and unseen feature weights. Because, if so, the objective function terms involving λ_C

are now $\frac{\lambda_C^2}{2\sigma^2} + \frac{\gamma}{2}(\lambda_A - \lambda_C)^2$. Due to the existence of l_2, its minimizer is no longer $\lambda_C = \lambda_A$, but rather $\lambda_C < \lambda_A$. Moreover, the value of λ_A will be pulled away from its expected value by FAR. If we assume that, in the labeled training data, feature A is mostly with the positive class and feature B is mostly with the negative class, then figure 4.2 illustrates the weights learned by (a) FAR with l_2 on seen features only, and (b) FAR with l_2 on all features. We observe in (b) that λ_C will be less than λ_A due to l_2's tendency to reduce λ_C, and λ_A is also decreased because A is connected to C. The reduction of λ_A has a negative effect on the training set error rate, as some positive-class samples will now be classified as negative class by mistake. Therefore the bias term, indicated by D, becomes positive to compensate the reduction of λ_A. These are not the desired behaviors of FAR. Hence, we should only use l_2 on seen feature weights.

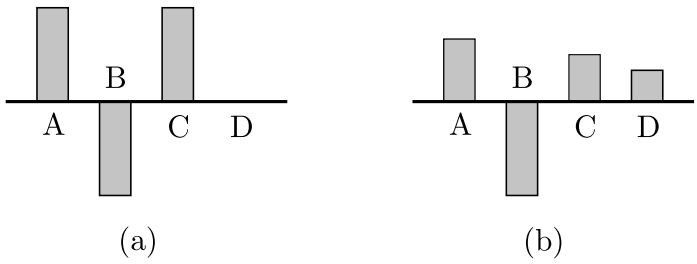
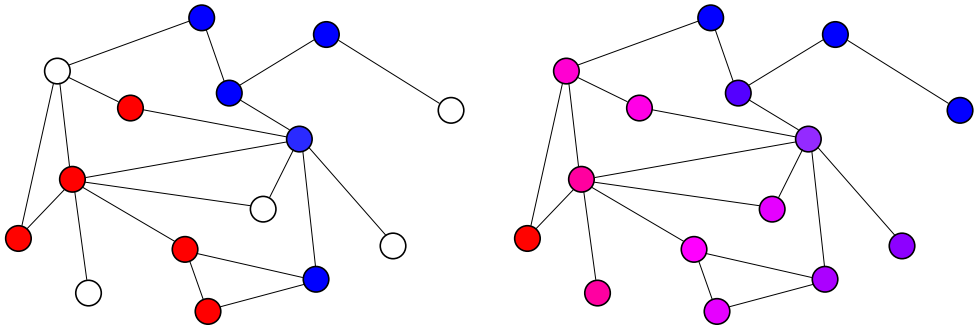


Figure 4.2: Weights learned by (a) FAR with no l_2 on feature C, and (b) FAR with l_2 on feature C. Note that D represents the bias term.

The second example, shown in figure 4.1(b), describes a case where the unseen feature C is connected to both seen features A and B. It may be that we do not have much direct evidence from unlabeled data that feature A and B are similar, but we are very confident that they are both similar to C. As a result, the objective function term that involves λ_C is $\frac{\gamma}{2}[(\lambda_A - \lambda_C)^2 + (\lambda_B - \lambda_C)^2]$, whose minimizer is $\lambda_C = \frac{1}{2}(\lambda_A + \lambda_B)$. Based on the feature affinity matrix, we expect λ_A and λ_B to be close, as they should be both close to λ_C. However, if, due to limited labeled training data, λ_A and λ_B differ too much, then FAR plays the role of pushing λ_A and λ_B closer through the link of λ_C. Therefore, the participance of unseen feature C not only gets λ_C a new value estimated from λ_A and λ_B, but also prevents learning improper λ_A and λ_B that contradict unlabeled data.

The feature affinity matrix learned from real unlabeled data is far more complicated than the ones shown in figure 4.1. Usually, a feature has connections to many other features, and the analysis is no longer as simple as above. The basic idea still applies, that is, FAR pushes the weights of similar features closer. As a contrived but illustrative example, figure 4.3 shows feature graphs and feature weights learned with FAR (a) and without FAR (b). The colored nodes in (a) indicate seen features, whereas white nodes in (a) indicate unseen features. The links in the graphs indicate feature similarity. The colors indicate feature weight, with red being positive and blue being negative. Comparing (b) to (a), with FAR, feature weights are more similar to each other if the features are similar, and unseen features are assigned weights similar to their connected neighboring features.



(a) (b)

Figure 4.3: Example feature graph with (a) and without (b) FAR

In general, we should avoid using a dense $\mathbf{W}$, i.e., a fully connected feature graph, which will lead to the undesirable $O(N^2)$ computation in FAR, since N is typically large for text classification problems. In many cases, a sparse $\mathbf{W}$ may be constructed, thus the computation complexity in FAR may be much less than $O(N^2)$.

It has to be pointed out that FAR only impacts the distance between feature weights for one class, rather than those belonging to different classes. That is to say, it may push λ_{ik} and λ_{jk} closer, but it never tries to push λ_{ik_1} and λ_{jk_2} closer, if $k_1 \neq k_2$. This can be considered as regularization on a per-class basis, where the regularization of different classes are independent. It is one step forward from l_2, which regularizes each feature

weight independently.

Unlike Bayesian networks [Jensen, 1996, Friedman et al., 1997] and Markov logic networks [Richardson and Domingos, 2006], the feature graph encoded by $\mathbf{W}$ only models feature similarities, and it does not involve class variables. Bayesian networks and Markov logic networks are graphical models that define probability distributions over features and class variables, while our feature graph does not have such a probabilistic interpretation and does not incorporate class variables in the graph. It is possible to derive feature similarities from feature probability distributions (see section 4.4), but since the result is generally not sparse, feature distributions are more useful in the context of feature cluster regularization, described next.

4.2 Feature Cluster Regularization

4.2.1 Definition

Instead of using a feature affinity matrix in regularization, we can also cluster features, and use the clustering information in regularization. Specifically, assume we have an $N \times C$ matrix $\mathbf{M}$ of feature-cluster (C clusters) memberships, whose element $m_{ic} \geq 0$ denotes the non-negative membership of feature i to cluster c. We call $\mathbf{M}$ the *feature-cluster matrix*. Then we can define feature cluster regularization that favors smaller within-cluster weight variances,

$$J_{\mathrm{FCR}}(\mathbf{\Lambda}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{c=1}^{C} \sum_{i=1}^{N} m_{ic} \left(\lambda_{ik} - \frac{\sum_{j=1}^{N} m_{jc} \lambda_{jk}}{\sum_{j=1}^{N} m_{jc}} \right)^2 . \tag{4.19}$$

Here, for simplicity, we also make an assumption that M satisfies

$$\sum_{i=1}^{N} \sum_{c=1}^{C} m_{ic} = 1, \tag{4.20}$$

which can be easily achieved by normalization of $\mathbf{M}$. As a result, m_{ic} can be considered as a probability mass function $p(f_i, c)$[1]. Therefore, $J_{\mathrm{FCR}}(\mathbf{\Lambda})$ is essentially the expected

[1]There are other choices of m_{ic} such as $m_{ic} = p(f_i|c)$ and $m_{ic} = p(c|f_i)$, each corresponding to the normalization constraint $\sum_{i=1}^{N} m_{ic} = 1$ and $\sum_{c=1}^{C} m_{ic} = 1$, respectively. But we think the feature-cluster joint probability model $p(i, c)$ is more general as it enables importance weighting on both features and clusters, while the conditional probabilities only allow for one of them. The conditional probability models can be considered as special cases of joint probability model when $p(c)$ or $p(f_i)$ is uniform, with the assumption that each cluster or feature is equally important in FCR.

weight variance of all clusters. Similar to the well-known formula $\mathrm{Var}(x) = \mathbb{E}[x^2] - \mathbb{E}^2[x]$ in probability theory, we can rewrite $J_{\mathrm{FCR}}(\mathbf{\Lambda})$ in the following form, which will be easier for the analysis of convexity and derivatives.

$$J_{\mathrm{FCR}}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\sum_{i=1}^{N} m_{ic}\left[\lambda_{ik}^2 + \frac{\left(\sum_{j=1}^{N} m_{jc}\lambda_{jk}\right)^2}{\left(\sum_{j=1}^{N} m_{jc}\right)^2} - \frac{2\lambda_{ik}\sum_{j=1}^{N} m_{jc}\lambda_{jk}}{\sum_{j=1}^{N} m_{jc}}\right] \tag{4.21}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left[\sum_{i=1}^{N} m_{ic}\lambda_{ik}^2 + \sum_{i=1}^{N} m_{ic}\frac{\left(\sum_{i=1}^{N} m_{ic}\lambda_{ik}\right)^2}{\left(\sum_{i=1}^{N} m_{ic}\right)^2} - \frac{2\sum_{i=1}^{N} m_{ic}\lambda_{ik}\sum_{j=1}^{N} m_{jc}\lambda_{jk}}{\sum_{j=1}^{N} m_{jc}}\right] \tag{4.22}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left[\sum_{i=1}^{N} m_{ic}\lambda_{ik}^2 - \frac{\left(\sum_{i=1}^{N} m_{ic}\lambda_{ik}\right)^2}{\sum_{i=1}^{N} m_{ic}}\right]. \tag{4.23}$$

Since we use a matrix to encode clustering, the clustering itself does not have to be hard cluster assignment. Hard cluster assignment corresponds to a sparse $\mathbf{M}$ where each row has only one non-zero element. Soft cluster assignment corresponds to a less sparse $\mathbf{M}$. Soft clustering is more general than hard clustering, and it is closer to how real world features are related — there is often no clear-cut distinction between two clusters. Although similarly motivated as FAR, FCR has the advantage that it only requires moderate computation of $O(NC)$ (compared to $O(N^2)$ for FAR with a dense $\mathbf{W}$) even if a dense $\mathbf{M}$ is used when $C \ll N$, which is common. Of course, a sparser $\mathbf{M}$ will further speed up computation, but at a cost of less encoded information.

It can be shown that $J_{\mathrm{FCR}}(\mathbf{\Lambda})$ is convex with respect to $\mathbf{\Lambda}$.

Proof. To prove its convexity, we only need to show that, given $N \times K$ real matrices $\mathbf{X}$ and $\mathbf{Y}$, for any $0 \leq \alpha \leq 1$, the following inequality holds.

$$J_{\mathrm{FCR}}(\alpha\mathbf{X} + (1-\alpha)\mathbf{Y}) \leq \alpha J_{\mathrm{FCR}}(\mathbf{X}) + (1-\alpha)J_{\mathrm{FCR}}(\mathbf{Y}). \tag{4.24}$$

Each side of this inequality can be computed as the following.

$$J_{\mathrm{FCR}}(\alpha\mathbf{X}+(1-\alpha)\mathbf{Y}) \tag{4.25}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left[\sum_{i=1}^{N} m_{ic}\left[\alpha x_{ik}+(1-\alpha)y_{ik}\right]^2 - \frac{\left(\sum_{i=1}^{N} m_{ic}\left[\alpha x_{ik}+(1-\alpha)y_{ik}\right]\right)^2}{\sum_{i=1}^{N} m_{ic}}\right] \tag{4.26}$$

$$\begin{aligned} &= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\Bigg\{\sum_{i=1}^{N} m_{ic}[\alpha^2 x_{ik}^2+(1-\alpha)^2 y_{ik}^2+2\alpha(1-\alpha)x_{ik}y_{ik}] \\ &\quad - \frac{1}{\sum_{i=1}^{N} m_{ic}}\Bigg[\alpha^2\left(\sum_{i=1}^{N} m_{ic}x_{ik}\right)^2+(1-\alpha)^2\left(\sum_{i=1}^{N} m_{ic}y_{ik}\right)^2 \\ &\quad + 2\alpha(1-\alpha)\left(\sum_{i=1}^{N} m_{ic}x_{ik}\right)\left(\sum_{i=1}^{N} m_{ic}y_{ik}\right)\Bigg]\Bigg\} \end{aligned} \tag{4.27}$$

$$\begin{aligned} &= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\Bigg\{\alpha^2\left[\sum_{i=1}^{N} m_{ic}x_{ik}^2 - \frac{\left(\sum_{i=1}^{N} m_{ic}x_{ik}\right)^2}{\sum_{i=1}^{N} m_{ic}}\right]+(1-\alpha)^2\left[\sum_{i=1}^{N} m_{ic}y_{ik}^2 - \frac{\left(\sum_{i=1}^{N} m_{ic}y_{ik}\right)^2}{\sum_{i=1}^{N} m_{ic}}\right] \\ &\quad + 2\alpha(1-\alpha)\left[\sum_{i=1}^{N} m_{ic}x_{ik}y_{ik} - \frac{\left(\sum_{i=1}^{N} m_{ic}x_{ik}\right)\left(\sum_{i=1}^{N} m_{ic}y_{ik}\right)}{\sum_{i=1}^{N} m_{ic}}\right]\Bigg\}. \end{aligned} \tag{4.28}$$

$$\alpha J_{\mathrm{FCR}}(\mathbf{X})+(1-\alpha)J_{\mathrm{FCR}}(\mathbf{Y}) \tag{4.29}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left\{\alpha\left[\sum_{i=1}^{N} m_{ic}x_{ik}^2 - \frac{\left(\sum_{i=1}^{N} m_{ic}x_{ik}\right)^2}{\sum_{i=1}^{N} m_{ic}}\right]+(1-\alpha)\left[\sum_{i=1}^{N} m_{ic}y_{ik}^2 - \frac{\left(\sum_{i=1}^{N} m_{ic}y_{ik}\right)^2}{\sum_{i=1}^{N} m_{ic}}\right]\right\}. \tag{4.30}$$

The difference between two terms is

$$J_{\mathrm{FCR}}(\alpha\mathbf{X}+(1-\alpha)\mathbf{Y})-[\alpha J_{\mathrm{FCR}}(\mathbf{X})+(1-\alpha)J_{\mathrm{FCR}}(\mathbf{Y})] \tag{4.31}$$

$$\begin{aligned}&=-\frac{\alpha(1-\alpha)}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\Bigg[\sum_{i=1}^{N}m_{ic}x_{ik}^2-\frac{\left(\sum_{i=1}^{N}m_{ic}x_{ik}\right)^2}{\sum_{i=1}^{N}m_{ic}}+\sum_{i=1}^{N}m_{ic}y_{ik}^2-\frac{\left(\sum_{i=1}^{N}m_{ic}y_{ik}\right)^2}{\sum_{i=1}^{N}m_{ic}}\\&\quad-2\sum_{i=1}^{N}m_{ic}x_{ik}y_{ik}+2\frac{\left(\sum_{i=1}^{N}m_{ic}x_{ik}\right)\left(\sum_{i=1}^{N}m_{ic}y_{ik}\right)}{\sum_{i=1}^{N}m_{ic}}\Bigg]\end{aligned} \tag{4.32}$$

$$=-\frac{\alpha(1-\alpha)}{2}\left[\sum_{i=1}^{N}m_{ic}(x_{ik}-y_{ik})^2-\frac{\left[\sum_{i=1}^{N}m_{ic}(x_{ik}-y_{ik})\right]^2}{\sum_{i=1}^{N}m_{ic}}\right] \tag{4.33}$$

$$=-\frac{\alpha(1-\alpha)}{2}J_{\mathrm{FCR}}(\mathbf{X}-\mathbf{Y}) \tag{4.34}$$

$$\leq 0. \tag{4.35}$$

□

The gradient of FCR can be computed as the following.

$$\frac{\partial J_{\mathrm{FCR}}(\mathbf{\Lambda})}{\partial\lambda_{ik}}=\sum_{c=1}^{C}m_{ic}\left(\lambda_{ik}-\frac{\sum_{j=1}^{N}m_{jc}\lambda_{jk}}{\sum_{j=1}^{N}m_{jc}}\right). \tag{4.36}$$

The gradient is proportional to the weighted difference between the feature weight and the cluster means. Therefore, it pushes the weights towards the cluster means.

Similar to FAR, when applied in MaxEnt training, the overall objective function is the following with l_2 only applied to seen feature weights,

$$J(\mathbf{\Lambda})=-\sum_{(\mathbf{x},y)\in\mathcal{L}}\log p(y|\mathbf{x})+\sum_{i=1}^{N_s}\sum_{k=1}^{K}\frac{\lambda_{ik}^2}{2\sigma^2}+\gamma J_{\mathrm{FCR}}(\mathbf{\Lambda}). \tag{4.37}$$

4.2.2 Analysis

It is possible to derive feature affinity based on feature clustering. As $m_{ic}=p(f_i,c)=p(f_i|c)p(c)$, we may define feature affinity as

$$w_{ij}=p(f_i,f_j)=\sum_{c=1}^{C}p(f_i|c)p(f_j|c)p(c). \tag{4.38}$$

It assumes that feature co-occurrence probabilities can be modeled by C mixtures. Under this assumption, it can be shown that FCR is equivalent to FAR. Both FAR and FCR will achieve the same result.

Proof.

$$J_{\mathrm{FAR}}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j,\ i<j} w_{ij}(\lambda_{ik}-\lambda_{jk})^2 \tag{4.39}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\sum_{i,j,\ i<j} p(f_i|c)p(f_j|c)p(c)\left(\lambda_{ik}-\lambda_{jk}\right)^2 \tag{4.40}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\sum_{i,j,\ i<j} p(f_i|c)p(f_j|c)p(c)\left(\lambda_{ik}^2+\lambda_{jk}^2-2\lambda_{ik}\lambda_{jk}\right) \tag{4.41}$$

$$= \frac{1}{4}\sum_{k=1}^{K}\sum_{c=1}^{C}\sum_{i,j} p(f_i|c)p(f_j|c)p(c)\left(\lambda_{ik}^2+\lambda_{jk}^2-2\lambda_{ik}\lambda_{jk}\right) \tag{4.42}$$

$$\begin{aligned}= \frac{1}{4}\sum_{k=1}^{K}\sum_{c=1}^{C} p(c)\Bigg[&\sum_j p(f_j|c)\sum_i p(f_i|c)\lambda_{ik}^2+\sum_i p(f_i|c)\sum_j p(f_j|c)\lambda_{jk}^2\\ &-2\sum_i p(f_i|c)\lambda_{ik}\sum_j p(f_j|c)\lambda_{jk}\Bigg]\end{aligned} \tag{4.43}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C} p(c)\left[\sum_i p(f_i|c)\lambda_{ik}^2-\left(\sum_i p(f_i|c)\lambda_{ik}\right)^2\right]. \tag{4.44}$$

$$J_{\mathrm{FCR}}(\mathbf{\Lambda}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left[\sum_{i=1}^{N} m_{ic}\lambda_{ik}^2-\frac{\left(\sum_{i=1}^{N} m_{ic}\lambda_{ik}\right)^2}{\sum_{i=1}^{N} m_{ic}}\right] \tag{4.45}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left[\sum_{i=1}^{N} p(f_i|c)p(c)\lambda_{ik}^2-\frac{\left(\sum_{i=1}^{N} p(f_i|c)p(c)\lambda_{ik}\right)^2}{\sum_{i=1}^{N} p(f_i|c)p(c)}\right] \tag{4.46}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C}\left[\sum_{i=1}^{N} p(f_i|c)p(c)\lambda_{ik}^2-\frac{\left(\sum_{i=1}^{N} p(f_i|c)\lambda_{ik}\right)^2}{\sum_{i=1}^{N} p(f_i|c)}p(c)\right] \tag{4.47}$$

$$= \frac{1}{2}\sum_{k=1}^{K}\sum_{c=1}^{C} p(c)\left[\sum_{i=1}^{N} p(f_i|c)\lambda_{ik}^2-\left(\sum_{i=1}^{N} p(f_i|c)\lambda_{ik}\right)^2\right] \tag{4.48}$$

$$= J_{\mathrm{FAR}}(\mathbf{\Lambda}). \tag{4.49}$$

□

From equation (4.41) to (4.42), we remove the constraint $i < j$ in the summation. Because $\mathbf{W}$ is symmetric, this effectively doubles the summation, which is accounted for by shrinking the scale factor by half.

Intuitively, equation (4.38) defines a low rank approximation to the full feature affinity matrix $\mathbf{W}$. This approximation can reduce the computational cost associated with a full $\mathbf{W}$, and it is effectively smoothing the co-occurrence counts. While smoothing offers the potential for more robust estimates and associated improved performance, the reduced degrees of freedom could alternatively lead to performance degradation. We explore this question in Chapter 5, where a framework with reduced computational cost is introduced.

4.3 Fixed Point Weight Propagation

When γ is very small, the impact of FAR/FCR to seen feature weights can be neglected. The optimization algorithm will find the same seen feature weights as obtained by a conventional l_2 MaxEnt trainer. In this case, the unseen feature weights may be found directly by minimizing $J_{\text{FAR}}(\mathbf{\Lambda})$ or $J_{\text{FCR}}(\mathbf{\Lambda})$, i.e., by setting the derivatives (4.18) and (4.36) to zero while treating the learned seen feature weights as given. This yields, for FAR,

$$\sum_{j=1}^{N} w_{ij}(\lambda_{ik} - \lambda_{jk}) = 0, \quad N_s < i \leq N, 1 \leq k \leq K, \tag{4.50}$$

and for FCR,

$$\sum_{c=1}^{C} m_{ic} \left(\lambda_{ik} - \frac{\sum_{j=1}^{N} m_{jc}\lambda_{jk}}{\sum_{j=1}^{N} m_{jc}} \right) = 0, \quad N_s < i \leq N, 1 \leq k \leq K. \tag{4.51}$$

Both equations above are K linear systems, each set with N_u equations and variables. Although these equations can be solved efficiently using linear algebra when N_u is small, it can be expensive when N_u is large, which is common. Therefore, we propose to use the following fixed point algorithm to solve them. Because it is very similar to label propagation [Zhu and Ghahramani, 2002] but it is applied on weights rather than labels, we call it *fixed point weight propagation* (FPWP), which is operated on a feature similarity graph.

The linear equations above can be rewritten as the following. For FAR,

$$\lambda_{ik} = \frac{\sum_{j=1}^{N} w_{ij}\lambda_{jk}}{\sum_{j=1}^{N} w_{ij}}, \quad N_s < i \leq N, 1 \leq k \leq K, \tag{4.52}$$

and for FCR,

$$\lambda_{ik} = \frac{1}{\sum_{c=1}^{C} m_{ic}} \sum_{c=1}^{C} \frac{\sum_{j=1}^{N} m_{ic}m_{jc}\lambda_{jk}}{\sum_{j=1}^{N} m_{jc}}, \quad N_s < i \leq N, 1 \leq k \leq K. \tag{4.53}$$

We can use these equations as update formula in the fixed point algorithm. Furthermore, these equations can be converted in a common matrix form,

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{4.54}$$

where for $1 \leq i \leq N_u$, $1 \leq j \leq N_u$, and $1 \leq k \leq K$,

$$x_i = \lambda_{i+N_s,k}, \tag{4.55}$$

$$a_{ij} = \begin{cases} \frac{w_{i+N_s,j+N_s}}{\sum_{n=1}^{N} w_{i+N_s,n}} & \text{for FAR,} \\ \frac{1}{\sum_{c=1}^{C} m_{i+N_s,c}} \sum_{c=1}^{C} \frac{m_{i+N_s,c}m_{j+N_s,c}}{\sum_{n=1}^{N} m_{nc}} & \text{for FCR,} \end{cases} \tag{4.56}$$

$$b_i = \begin{cases} \frac{\sum_{n=1}^{N_s} w_{in}\lambda_{nk}}{\sum_{n=1}^{N} w_{in}} & \text{for FAR,} \\ \frac{1}{\sum_{c=1}^{C} m_{i+N_s,c}} \sum_{c=1}^{C} \frac{\sum_{n=1}^{N_s}\sum_{c=1}^{C} m_{ic}m_{nc}\lambda_{nk}}{\sum_{n=1}^{N} m_{nc}} & \text{for FCR,} \end{cases} \tag{4.57}$$

and therefore the universal update equation is

$$\mathbf{x}^{(n+1)} = \mathbf{A}\mathbf{x}^{(n)} + \mathbf{b}. \tag{4.58}$$

Clearly, when there are clusters of unseen features that are not connected to any seen features, setting these unseen feature weights to any value that is consistent within each cluster will minimize the objective function. In our implementation, these unseen feature weights are initialized to zero and will stay at zero. Hence their analysis is trivial. By excluding their feature weights from (4.54), we can always obtain a more interesting but still general case, where any unseen feature is directly or indirectly connected to one or more seen features. Here, we show that if (4.54) has one solution (fixed point), we can converge to it by using (4.58). This proof is based on the one in [Zhu and Ghahramani, 2002] with more general cases considered.

Proof. Based on the iterative updates using (4.58), we can calculate $\mathbf{x}^{(n)}$ using

$$\mathbf{x}^{(n)} = \mathbf{A}^n \mathbf{x}^{(0)} + \sum_{i=0}^{n-1} \mathbf{A}^i \mathbf{b}. \tag{4.59}$$

From the definition of a_{ij}, we can see that each row of $\mathbf{A}$ satisfies

$$S_i(\mathbf{A}) \equiv \sum_{j=1}^{N_u} a_{ij} \leq 1, \quad 1 \leq i \leq N_u. \tag{4.60}$$

If the inequality is strict for all i, e.g., $\sum_{j=1}^{N_u} a_{ij} < 1$, then $\forall i$ there exists $\alpha < 1$, such that $\sum_{j=1}^{N_u} a_{ij} < \alpha$. This corresponds to a feature graph where any unseen feature is directly connected to at least one seen feature. Assume the row sums of $\mathbf{A}^n$, $S_i(\mathbf{A}^n)$, satisfies $S_i(\mathbf{A}^n) < \alpha^n$, by induction, the row sums of $\mathbf{A}^{n+1}$ can be calculated recursively,

$$S_i(\mathbf{A}^{n+1}) = \sum_{j=1}^{N_u} a_{ij} S_j(\mathbf{A}^n) < \alpha^n \sum_{j=1}^{N_u} a_{ij} < \alpha^{n+1}. \tag{4.61}$$

Therefore, $\lim_{n\to\infty} S_i(\mathbf{A}^n) = 0$, thus $\lim_{n\to\infty} \mathbf{A}^n \mathbf{x}^{(0)} = 0$, which means that $\mathbf{x}^{(n)}$ converges regardless of the initial value $\mathbf{x}^{(0)}$.

On the other hand, there may exist a set of rows in $\mathbf{A}^n$, indexed by $i \in \mathcal{I}_n$, such that the corresponding row sum $S_i(\mathbf{A}^n) = 1$. This corresponds to the case where some unseen features are not directly connected to any seen features. We will show that $\mathcal{I}_{n+1} \subset \mathcal{I}_n$, i.e., the number of unity-sum rows necessarily decreases as n increases. First of all, for $i \notin \mathcal{I}_n$,

$$S_i(\mathbf{A}^{n+1}) = \sum_{j=1}^{N_u} a_{ij} S_j(\mathbf{A}^n) \leq \sum_{j=1}^{N_u} a_{ij} < 1. \tag{4.62}$$

Hence, the row sums of rows not in $\mathcal{I}_n$ remain less than one, thus $\mathcal{I}_{n+1} \subseteq \mathcal{I}_n$. We only need to show that the sum of at least one row in $\mathcal{I}_n$ will become less than one after multiplied by $\mathbf{A}$. It can be shown by contradiction, because if $\forall i \in \mathcal{I}_n$, $S_i(\mathbf{A}^{n+1}) = 1$, then

$$S_i(\mathbf{A}^{n+1}) = \sum_{j=1}^{N_u} a_{ij} S_j(\mathbf{A}^n) = 1. \tag{4.63}$$

This could only happen when $a_{ij} = 0, \forall j \notin \mathcal{I}_n$. Since $\mathcal{I}_n \subseteq \mathcal{I}_1$, we have $a_{ij} = 0, \forall j \notin \mathcal{I}_1$, i.e., the features in $\mathcal{I}_n$ are in isolated clusters neither directly nor indirectly connected to any seen features, which has been precluded.

Because $\mathcal{I}_{n+1} \subset \mathcal{I}_n$, it takes at most N_u multiplications before the row sums become less than one. The row sums of $\mathbf{A}^{N_u}$ must be less than one. Therefore, $\lim_{n\to\infty} S_i(\mathbf{A}^{nN_u}) = 0$, thus $\lim_{n\to\infty} \mathbf{A}^{nN_u}\mathbf{x}^{(0)} = 0$, which again means that $\mathbf{x}^{(n)}$ converges regardless of the initial value $\mathbf{x}^{(0)}$.

□

From the proof, we can see that FPWP tends to converge more slowly if not all the unseen features are directly connected to any seen features. In an extreme case, a remote unseen feature that is n links away from the seen features needs n iterations to propagate any change of seen feature weights to it. For example, in FAR (figure 4.4), it takes n iterations to propagate the change of seen feature S to the furthest unseen feature U_n.

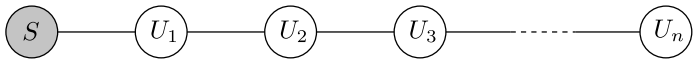


Figure 4.4: Feature graph: unseen feature U_n is indirectly connected to seen feature S through $n-1$ intermediate unseen features

FPWP is not only useful when γ is small. We find that, even when γ is not small, the unseen feature weights converge more slowly than the seen feature weights do. This is because seen feature weights receive gradients from both the loglikelihood term and the regularization term, while the unseen feature weights only receive gradients from the regularization term. Usually γ is set to a value such that the regularization term can play a role in learning unseen feature weights, but not strong enough to override the loglikelihood term. Hence the convergence of unseen feature weights is slower than that of seen feature weights. This is also problematic when a seen feature has a large fanout. As the gradients of the unseen feature weights are proportional to their difference to the cluster mean, which is dominated by the unseen features, the unseen feature weights and cluster mean tend to move away from zero very slowly.

To overcome these deficiencies, we propose to use FPWP alternating with gradient update.

In algorithm 5, seen feature weights are updated by L-BFGS, and the unseen feature

Algorithm 5: MaxEnt training with FPWP

```
Initialize all feature weights and bias terms to zero;
while not converged do
    Run one-step L-BFGS update on seen feature weights and bias terms;
    Treat seen feature weights as fixed, run FPWP to update unseen feature weights;
end
```

weights are updated by FPWP. These two steps are applied alternatingly, such that the change of seen feature weights can immediately affect the next update of unseen feature weights, and vise versa. From equations (4.2) and (4.37), we can see that when γ is close to zero, the FAR/FCR term has negligible impact on the overall objective function and its gradient. Therefore, alternating updating of seen and unseen feature weights is similar to updating these weights separately, i.e., we learn seen feature weights using a conventional MaxEnt without FAR/FCR, followed by FPWP using FAR/FCR. It is no longer the case when γ is large.

4.4 Learning Feature Affinity

Learning feature affinity amounts to learning similarity of features from unlabeled data. Although word similarity may be derived from various knowledge bases such as WordNet, we propose to learn it from data. The reason is because word features are only one type of features; it would be more useful we can use data-driven methods to learn similarity of features not limited to words. Moreover, we would like to have methods that can generalize to languages with few resources. Data-driven word similarity estimation methods have been studied in [Dagan et al., 1999, Terra and Clarke, 2003], and most of them can be roughly categorized into two classes: co-occurrence-based and context-based. In co-occurrence-based methods, co-occurrence statistics of pairs of words are the basic information to determine the similarities of word pairs. In context-based methods, the distribution of words within a certain context of the current word is considered as a property of the current word, and the difference between a pair of distributions is used to compute the similarity of the correspond-

ing word pair. The computational complexity of context-based methods is higher than that of co-occurrence-based methods, as we need to, for each word, compute the distribution of words in its context, and, for each word pair, compute the difference of distributions. This has problems if we need to increase feature dimensionality, which is the size of the vocabulary. For co-occurrence-based methods, we do not need to explicitly compute the similarity for all word pairs, but rather only for the pairs of words that have co-occurred. The pairs of words that have not co-occurred will have a similarity of zero. Therefore, we only use co-occurrence-based methods in this work. These methods do not limit features to words, but in the discussion to follow, we refer to features as being associated with words to simplify the discussion.

In the following sections, we introduce five methods to learn feature affinity a_{ij} between feature i and feature j, including co-occurrence probability (COOC), normalized co-occurrence probability (NCOOC), Jaccard index (JAC), Dice coefficient (DICE), and non-negative point-wise mutual information (NPMI).

4.4.1 Co-occurrence Probability

This method is based on the assumption that feature affinity between words i and j is proportional to the probability for words i and j to co-occur, e.g., $a_{ij}^{\text{COOC}} = p(x_i > 0 \wedge x_j > 0)$. Note here $x_i > 0$ essentially means that word i appears in a sample. For binary word occurrence features, $x_i = 1$ if word i appears in a sample. For word count features, $x_i = n > 0$ if word i appears n times. For continuous value features such as TF-IDF, $x_i > 0$ if word i appears. The use of $x_i > 0$ covers all the above cases. The unit of text in which we consider co-occurrence is the same as the unit of classification. For example, if the text classification is document-level, we consider document-level co-occurrence. If the text classification is sentence-level, we consider sentence-level co-occurrence.

This co-occurrence probability-based feature affinity can be estimated using maximum likelihood by

$$a_{ij}^{\text{COOC}} = \frac{\#\text{ samples where } x_i > 0 \text{ and } x_j > 0}{\#\text{ all samples}}. \tag{4.64}$$

The range of affinity values is thus $[0, 1]$, with 1 indicating the largest similarity.

4.4.2 Normalized Co-occurrence Probability

The problem with co-occurrence probability is that, if two words both appear frequently but are not correlated, their affinity estimated by co-occurrence probability may still be high as the probability of chance co-occurrence is high. Normalized co-occurrence probability tries to alleviate this problem by normalizing the co-occurrence probability by individual probabilities. It can be computed by

$$a_{ij}^{\text{NCOOC}} = \frac{p(x_i > 0 \wedge x_j > 0)}{\sqrt{p(x_i > 0)}\sqrt{p(x_j > 0)}}, \tag{4.65}$$

where the numerator is computed by co-occurrence probability as in (4.64), and the denominator is estimated by

$$p(x_i > 0) = \frac{\#\text{ samples where } x_i > 0}{\#\text{ all samples}}. \tag{4.66}$$

The range of a_{ij}^{NCOOC} is still $[0, 1]$, with 1 indicating the largest similarity. Note that to achieve the largest similarity, both features must always co-occur, but, unlike co-occurrence probability, they do not need to appear in all samples.

4.4.3 Jaccard Index

The Jaccard coefficient measures similarity between sample sets, A and B, and is defined as the size of the intersection divided by the size of the union of the sample sets,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{4.67}$$

where $|\cdot|$ computes the cardinality of a set.

If we define A and B as sets of samples where word i and j appears, respectively,

$$A = \{\mathbf{x}|x_i > 0\}, \tag{4.68}$$

$$B = \{\mathbf{x}|x_j > 0\}, \tag{4.69}$$

then $J(A, B)$ is a measure of affinity between word i and j. The range of affinity values calculated in this way is $[0, 1]$.

4.4.4 Dice Coefficient

Dice coefficient is also a similarity measure between sets, and it is closely related to Jaccard index. It is defined as, for sets A and B,

$$D(A,B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{4.70}$$

The range of affinity values calculated in this way is $[0, 1]$.

4.4.5 Non-negative Point-wise Mutual Information

For two random variables, X and Y, mutual information is defined as

$$I(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}, \tag{4.71}$$

but it is not a suitable similarity measure for features. To see that, imagine that there are two binary features i and j. For half of the samples, $x_i = 1,\ x_j = 0$, and for another half, $x_i = 0,\ x_j = 1$. Namely, one word is always present when the other is not. Then the mutual information between both features is high, since the presence of one feature necessarily indicates the absence of the other feature, which is counter-intuitive as a similarity measure. Therefore, instead of computing the expectation of $\log \frac{p(x_i,x_j)}{p(x_i)p(x_j)}$, we just evaluate it at one point where $x_i = 1$ and $x_j = 1$. That value can be used to compute feature affinity.

According to probability theory, if $p(x_i > 0, x_j > 0) = p(x_i > 0)p(x_j > 0)$, then events $x_i > 0$ and $x_j > 0$ are independent, which should correspond to minimal affinity between words i and j. On the other hand, if $p(x_i > 0, x_j > 0) > p(x_i > 0)p(x_j > 0)$, then words i and j are co-occurring more than chance, which should lead to high affinity. Based on the same intuition, it seems that we should assign negative affinity to words i and j if they co-occur less than chance, e.g., $p(x_i > 0, x_j > 0) < p(x_i > 0)p(x_j > 0)$, but negative affinity is not permitted as it destroys the convexity of the objective function. Hence, we assign zero affinity to that case, keeping affinity values non-negative. The formula for computing non-negative point-wise mutual information is

$$a_{ij}^{\text{NPMI}} = \max\left(0, \log \frac{p(x_i > 0 \wedge x_j > 0)}{p(x_i > 0)p(x_j > 0)}\right). \tag{4.72}$$

In theory, with unlimited data, the range of affinity values calculated in this way is $[0, +\infty)$. The unbounded value is achieved when the frequencies of words x_i and x_j approach zero while x_i and x_j still always co-occur. In practice, the minimum non-zero probability of a word is the reciprocal of the number of unlabeled data samples $|\mathcal{U}|$. Therefore $a_{ij}^{\text{NPMI}} \leq \log |\mathcal{U}|$. This upper bound is achieved only when two words appear once and together in the unlabeled data, which is extremely rare. We do not place any cap on the affinity values. This does not appear to be a problem in our experiments, as we usually put a cut-off on the minimum co-occurrence rate, which in turn sets a lower limit on word frequencies.

4.5 Learning Feature Clustering

As we mentioned, the feature cluster matrix $\mathbf{M}$ used in FCR is an $N \times C$ matrix, therefore it is not restricted to hard clustering only. If features are words, there are existing algorithms to cluster words from unlabeled text. In the following sections, we will introduce the some algorithms to derive $\mathbf{M}$, including Brown-style word clustering (Brown), probabilistic latent semantic analysis (PLSA) and latent Dirichlet allocation (LDA). Brown clustering is a context-based hard clustering and is aimed at word prediction, while PLSA and LDA are soft clusterings and are targeted to topic modeling. Note that one can always convert a soft clustering to a hard clustering, by picking the mostly associated cluster for each feature, but information is lost during this process.

4.5.1 Brown-style Word Clustering

The word clustering algorithm introduced by Brown et al. [1992] is an agglomerative clustering algorithm to generate word classes for class-based n-gram language models. It builds a bigram language model of word classes from the text data, and, in each iteration, merges the best pair of word classes that reduces the likelihood of the bigram language model the least. It is also equivalent to greedily merging word classes such that the reduction of mutual information between adjacent word classes is minimal. Although it was designed for class-based language modeling, its intuition applies well to our scenario. Simply put, if two words are often followed or preceded by similar words, then they are likely to be in the same cluster. Because this clustering algorithm has to be run for $V - C$ iterations, where V is

the vocabulary size and C is the number of clusters, and in each iteration it has to search for all word pairs, its running time is usually quite long.

The feature-clustering matrix is computed by

$$m_{ic}^{\text{Brown}} = \begin{cases} 1, & \text{if word } i \text{ belongs to cluster } c, \\ 0, & \text{otherwise.} \end{cases} \tag{4.73}$$

Because this is a hard clustering approach, the resulting $\mathbf{M}$ has only N non-zero elements, which has the advantage that the computation of the FCR regularizer is very efficient.

4.5.2 Probabilistic Latent Semantic Analysis

PLSA was introduced by Hofmann [1999] to model co-occurrence data, such as text data. It assumes that the document-word co-occurrence is generated by a mixture model

$$p(d, w) = \sum_z p(d|z)p(w|z)p(z), \tag{4.74}$$

where d is a document, w is a word, and z is a latent variable (cluster ID). The model parameters $p(d|z)$, $p(w|z)$, and $p(z)$ can be estimated by EM, and the feature-clustering matrix can be derived from

$$m_{ic}^{\text{PLSA}} = p(w = i\text{th word in vocabulary}|z = c)p(z = c). \tag{4.75}$$

$p(d|z)$ is not used in the feature-clustering matrix. Clearly, the resulting clustering is a soft clustering.

4.5.3 Latent Dirichlet Allocation

Blei et al. [2003] introduced LDA, a Bayesian framework for topic modeling. In this model, each word w_{id} in a text document d has a topic label z_{id} drawn from a multinomial distribution θ_i with a Dirichlet prior, and the actual word w_{id} is drawn from a multinomial distribution ϕ_z conditioned on the topic label z. Specifically, for each document d,

$$\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha}), \tag{4.76}$$

for each topic z,

$$\boldsymbol{\phi}_z \sim \text{Dir}(\boldsymbol{\beta}), \tag{4.77}$$

and for each word w_{id} in document d,

$$z_{id} \sim \text{Multinomial}(\boldsymbol{\theta}_d), \tag{4.78}$$

$$w_{id} \sim \text{Multinomial}(\boldsymbol{\phi}_z). \tag{4.79}$$

Here $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are concentration parameters for Dirichlet distribution, and we use $\mathbf{1}$ (flat Dirichlet distributions) for both throughout our experiments.

Gibbs sampling is commonly used to perform inference in this model, we can thus generate samples of topic labels for each word in the data, from which the number of times each word occurs with each topic can be counted. If we treat the topics as clusters, we are able to compute the feature-clustering matrix by Monte Carlo expectation,

$$m_{ic}^{\text{LDA}} = \frac{\#\text{ Gibbs samples where feature } i \text{ is assigned cluster } c}{\#\text{ Gibbs samples}}. \tag{4.80}$$

The resulting clustering is a soft clustering, since a word may be assigned different clusters in different Gibbs samples.

4.6 Experiments

4.6.1 Data and Tasks

We choose two types of tasks, e.g., sentiment classification and topic classification, to evaluate FAR/FCR, primarily due to the fact that researchers have reported success using only word features, and the algorithms introduced in this chapter are not scalable to high-dimensional features. We use the movie review data set for sentiment classification, and the 20 Newsgroups data set and Reuters-21578 data set for topic classification. We use the unlabeled versions of training, development and evaluation data to learn $\mathbf{W}$ and $\mathbf{M}$, as in transductive learning.

Movie Review Classification

The task of movie review classification was introduced in [Pang et al., 2002] as an application of sentiment analysis. The classification is done on the document-level in that paper, which

may be problematic as there might be both positive and negative sentences in the same review document. Therefore, the same authors published a sentence-level annotated movie review data set in [Pang and Lee, 2004]. This data set is comprised of 5000 positive and 5000 negative sentences extracted from `www.rottentomatoes.com` reviews.

We use ten-fold cross validation in our experiments, and we keep the training-development-evaluation ratio to 8:1:1. The results reported are classification accuracies. Feature affinity values are estimated based on co-occurrence within the unit of classification, which is the sentence in this case.

20 Newsgroups Topic Classification

The 20 newsgroups data set [Rennie, 2008] is a collection of approximately 20,000 newsgroup documents, partitioned nearly evenly across 20 different newsgroups as classes. We employ the "bydate" version of the data split as recommended by the author.

This data set has been divided into training and test sets, with 11314 training documents and 7532 test documents. When downsampling the training set for semi-supervised learning, we also create a development set using the data not sampled for training, and the size of the development set is about 10% of the original training set. The co-occurrence counts are estimated within documents.

TF-IDF features are extracted from each document, as is typically done in the topic categorization literature. In this task, we attempt to recover the newsgroup that the test documents came from using 20-class classification. We evaluate the performance using classification accuracy.

Reuters-21578 Topic Classification

The Reuters-21578 data set, compiled by Lewis [1999], contains 21578 news articles appeared on the Reuters newswire in 1987, each labeled by zero or more topic labels from 90 topics. It is a multi-label classification problem, as the same document may be classified to belong to multiple topic categories. The class distribution is not balanced.

We use the "ModApte" split of the data set, with 9603 training documents and 3299

test documents. When downsampling the training set for semi-supervised learning, we also create a development set using the data not sampled for training, and the size of the development set is about 10% of the original training set. Feature affinity values are estimated based on co-occurrence within documents.

TF-IDF features are commonly used for this task. As in the standard evaluation procedure for this data set [Joachims, 1997, 1998], we perform 90 binary classifications, one for each topic, and then compute the micro-averaged precision-recall breakeven point. The higher the breakeven point, the better the classification performance.

4.6.2 Experiment Paradigm

To compare the performance FAR/FCR with other related approaches, we use the following baselines.

$\mathbf{l_2}$ This is the conventional supervised MaxEnt with l_2 regularization. It is trained on the labeled training data only using the objective function defined in (2.2).

Self This approach uses a l_2 MaxEnt model in self-training. Initially, we train an l_2 MaxEnt model using only the labeled training data, and use it to predict the labels of the unlabeled data. Then we pick the unlabeled samples with the most confident predictions, that is, $p(y = \text{prediction}|\mathbf{x})$ is larger than a threshold θ, and add these samples to the labeled training data pool. We iterate this process until no more unlabeled samples can be confidently predicted.

XR This is expectation regularization proposed by Mann and McCallum [2007], where the regularizer pushes the predicted label distribution of the unlabeled data closer to some prior distribution, which can be set to the labeled data distribution.

For FAR, we experiment with the five feature affinity learning methods introduced in section 4.4, i.e., co-occurrence probability (COOC), normalized co-occurrence probability (NCOOC), Jaccard index (JAC), Dice coefficient (DICE), and non-negative point-wise mutual information (NPMI). For FCR, we experiment with the three feature clustering learning methods introduced in section 4.5, i.e., Brown-style word clustering (Brown), PLSA

and LDA. PLSA parameters are learned using EM of 1000 iterations. LDA parameters are learned using 1000 iterations of Gibbs sampling, with the first 500 iterations discarded as burn-in. To study the impact of labeled training data size on our algorithms, we down-sample the labeled training data to form a training set. We also sample a development set from the labeled data to perform hyper-parameter tuning. After the tuning is done, we use the optimal hyper-parameter to train a new model with the combination of training and development sets. We choose downsampling ratios 5% (training) + 5% (development), 10% (training) + 10% (development), and 40% (training) + 10% (development), leading to overall downsampling ratios of 10%, 20%, and 50%. For all these downsampling ratios, $\mathbf{W}$ and $\mathbf{M}$ are still learned from all the unlabeled data, including the sampled training data and test data with labels discarded. The scale factor γ that controls the strength of the FAR/FCR is tuned on the development set, by picking the best performing γ from a wide range of $10^{-10}, 10^{-9}, \ldots, 10^{10}$. Note that, at $\gamma = 10^{-10}$, FAR/FCR will not change the seen feature weights because their impacts are so small, but FPWP from seen features to unseen features still applies. The same scale factor tuning is applied to XR as well. Additional tuning for FCR is done on the number of clusters C, which is chosen from 10, 100, and 1000. For self-training, we tune the confidence threshold θ from 0.9 to 1.0 with a step size of 0.01.

Our implementation of FAR/FCR is based on Mallet [McCallum, 2002], a Java toolkit for machine learning and language processing. It has its own implementations of L-BFGS and MaxEnt, and we modify the objective computation and gradient calculation to allow for FAR/FCR. To use it, we provide a scale factor and a sparsely formatted $\mathbf{W}$ or $\mathbf{M}$ stored in a file to enable FAR/FCR. The learning of $\mathbf{W}$ and $\mathbf{M}$ is performed off-line. Due to the storage requirements and the need for efficiency, for 20 Newsgroups and Reuters-21578 topic classification, we truncate the co-occurrence counts that are less than five. This is very effective in increasing the sparsity of $\mathbf{W}$, as co-occurrence counts have a long tail. After truncation, the coverage of words in $\mathbf{W}$ of the entire vocabulary is 18% for 20 Newsgroups and 27% for Reuters-21578. For movie review classification, where the vocabulary size is small, we are able to keep all the non-zero co-occurrence counts.

Finally, we also design an experiment to show that FPWP is beneficial. In this experiment, we use the movie review data. Rather than cross-validation, we just sample 10% of the data as training data. $\mathbf{W}$ is learned from the entire data set using normalized co-occurrence probability. We first run training without FPWP using FAR with $\gamma = 0.01$, recording the objective function value and changes in model parameters in each iteration. The parameter change at iteration n is done using the following formula,

$$\Delta_n(\mathbf{\Lambda}) = \sqrt{\sum_{k=1}^{K} \sum_{i \in \mathcal{I}} \left(\lambda_{ik}^{(n)} - \lambda_{ik}^{(n-1)} \right)^2}, \tag{4.81}$$

where $\mathcal{I}$ is the set of indices of the features under consideration. We use two different configurations of $\mathcal{I}$:

Seen features where we have $\mathcal{I} = \{1, 2, \ldots, N_s\}$, and

Unseen features where we have $\mathcal{I} = \{N_s + 1, N_s + 2, \ldots, N\}$.

By monitoring the change of weights of both sets of features, we can better understand the impact of FPWP. After training without FPWP, a similar procedure is carried out for training with FPWP, and we compare how the objective function value and parameter changes vary across these two settings.

4.6.3 Results

For ease of visualization, we present the results for FAR and FCR separately. For FAR, the experimental results are shown in figure 4.5 (movie review classification), figure 4.6 (20 Newsgroups topic classification), and figure 4.7 (Reuters-21578 topic classification). For FCR, the experimental results are shown in figure 4.8 (movie review classification), figure 4.9 (20 Newsgroups topic classification), and figure 4.10 (Reuters-21578 topic classification).

Because tuning of FAR/FCR hyper-parameter γ is necessary, we also list the optimal hyper-parameters for different overall downsampling ratios in tables 4.1 – 4.6.

We test the statistical significance of the results using paired t-test. The FAR/FCR results are compared to the best baseline results. If the $p \leq 0.01$, then we denote the corre-

sponding result with two stars (**). If $0.01 < p \leq 0.05$, then we denote the corresponding result with one star (*). Otherwise, no star is shown.

The experimental results for assessing FPWP in the context of FAR are shown in figure 4.12 (objective function value for each iteration) and figure 4.11 (sum of squared feature weight change for each iteration).

We first summarize our observations in FAR experiments. In movie review classification, FAR with normalized co-occurrence probability achieves the best result (figure 4.5). Its largest gain over the best baseline is 2.7% absolute. All FAR experiments are able to achieve significant gains over the best baseline. In 20 Newsgroups topic classification, there is no all-around winner for different downsampling ratios (figure 4.6). FAR with Jaccard index, co-occurrence probability, and normalized co-occurrence probability are the best performers for 10%, 20%, and 50% overall downsampling ratios, respectively. All FAR improvements over the best baseline are significant. In Reuters-21578 topic classification, only the FAR improvements at 10% overall sampling ratio are significant (figure 4.7).

In tables 4.1 – 4.3, we observe that, in general, the hyper-parameter γ increases as the overall downsampling ratio increases. This is intuitive, because the impact of loglikelihood term in the objective function increases as more labeled training data is added. γ controls the strength of FAR, and it needs to be increased to keep up with the greater impact of the loglikelihood term. However, due to noise in data and discrete tuning step sizes, we see occasional drop of γ even though we add more labeled training data, especially in 20 Newsgroups and Reuters-21578 topic classification (table 4.2 and table 4.3).

It should be pointed out that the feature affinity learning methods we used are all based on a general co-occurrence function and are computationally cheap on the data sets in use. This is different from prior work, where the method has to be tailored to a particular task.

For FCR, PLSA achieves significant improvements in movie review classification for all downsampling ratios (figure 4.8). It also wins 20 Newsgroups topic classification for all downsampling ratios, where both PLSA and LDA obtain significant improvements (figure 4.9). No FCR methods obtain significant improvements in Reuters-21578 topic classification (figure 4.10). PLSA and LDA outperform Brown-style clustering consistently, with Brown-style clustering often being close to the baselines. This may be explained by the fact that

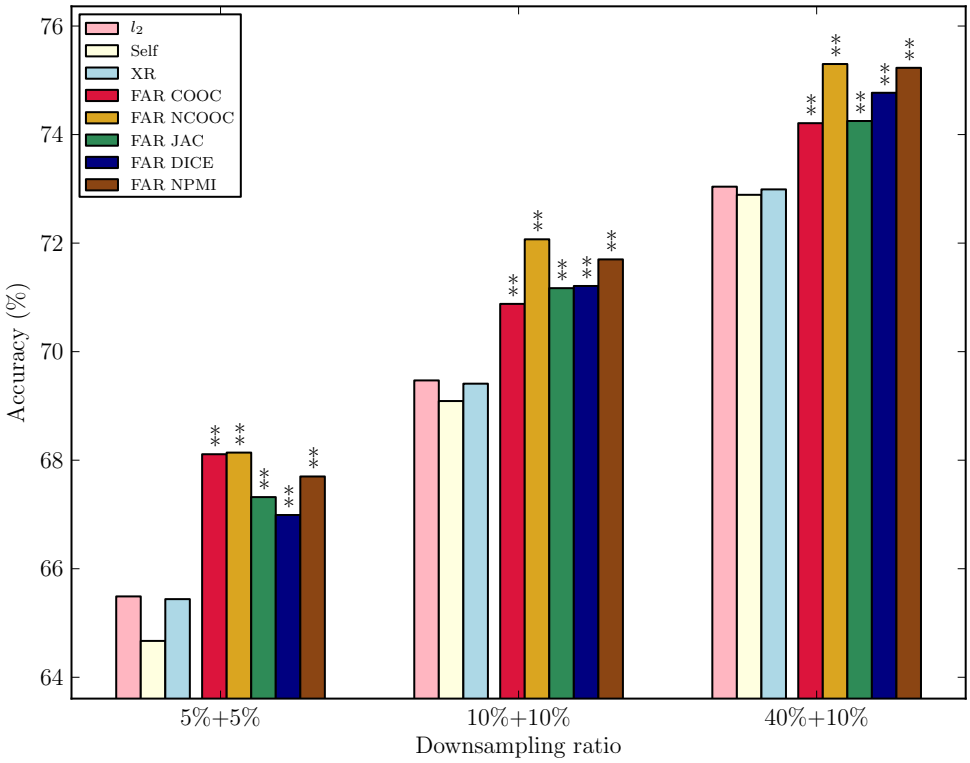


Figure 4.5: Movie review sentiment classification results (FAR)

Affinity	Downsampling ratio		
	10%	20%	50%
COOC	1	1	10
NCOOC	0.01	0.1	1
JAC	0.1	0.1	0.1
DICE	0.01	0.1	0.1
NPMI	0.0001	0.001	0.01

Table 4.1: Movie review FAR optimal hyper-parameters

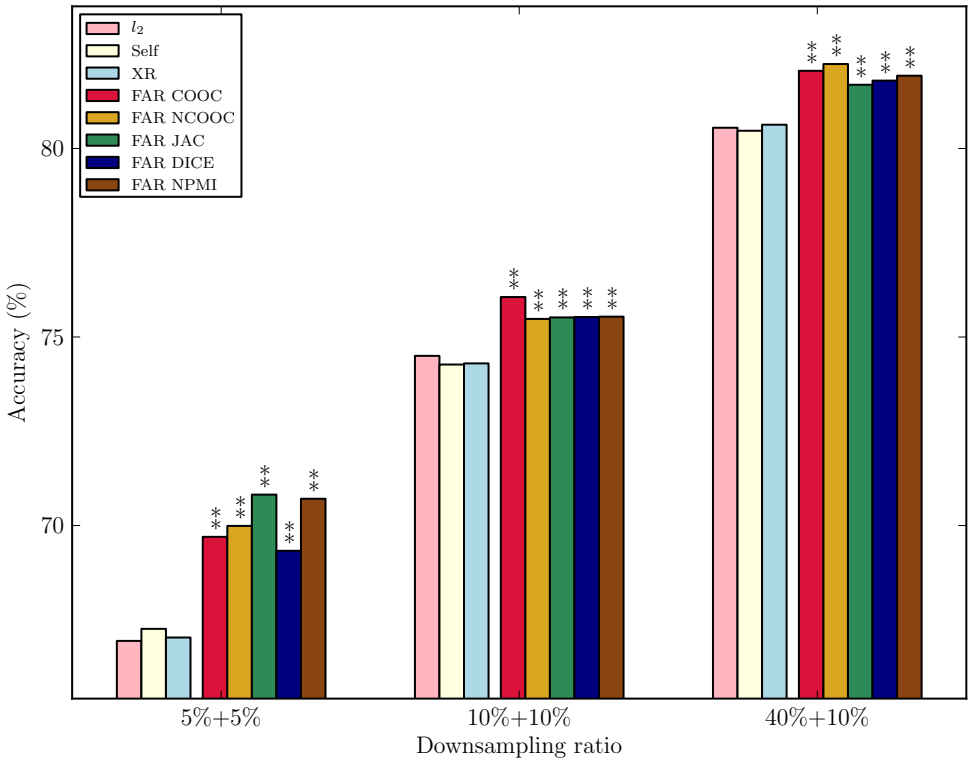


Figure 4.6: 20 Newsgroups topic classification results (FAR)

Affinity	Downsampling ratio		
	10%	20%	50%
COOC	0.01	0.01	0.1
NCOOC	0.0001	10^{-5}	0.001
JAC	0.001	10^{-6}	0.01
DICE	10^{-6}	10^{-6}	0.001
NPMI	10^{-5}	10^{-10}	0.0001

Table 4.2: 20 Newsgroups FAR optimal hyper-parameters

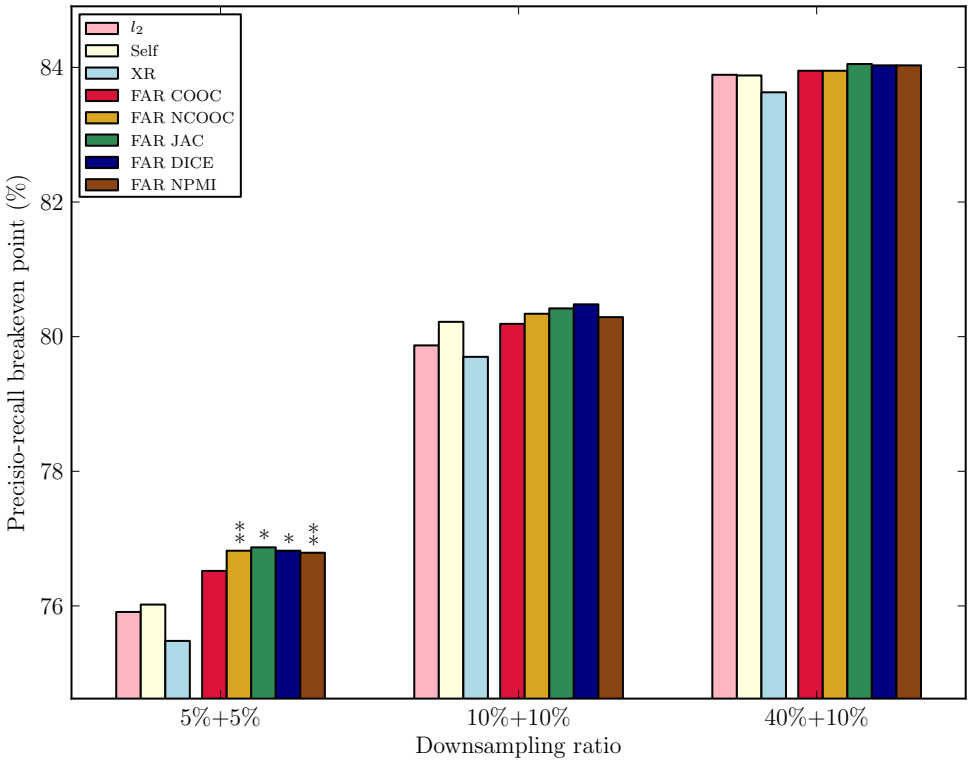


Figure 4.7: Reuters-21578 topic classification results (FAR)

Affinity	Downsampling ratio		
	10%	20%	50%
COOC	10^{-5}	0.001	10^{-6}
NCOOC	10^{-10}	10^{-7}	10^{-6}
JAC	10^{-10}	10^{-10}	10^{-6}
DICE	10^{-9}	10^{-10}	10^{-10}
NPMI	10^{-10}	10^{-6}	10^{-10}

Table 4.3: Reuters-21578 FAR optimal hyper-parameters

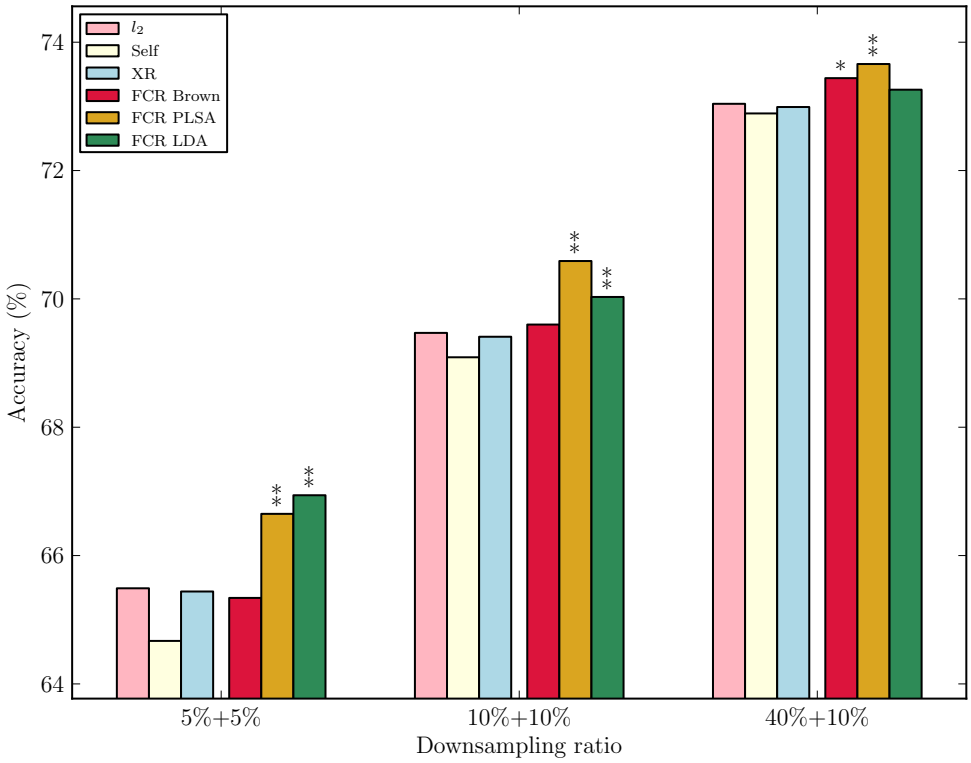


Figure 4.8: Movie review classification results (FCR)

Clustering	Downsampling ratio		
	10%	20%	50%
Brown	10^{-5}	0.01	1
PLSA	100	1000	1000
LDA	0.001	0.001	0.001

Table 4.4: Movie review FCR optimal hyper-parameters

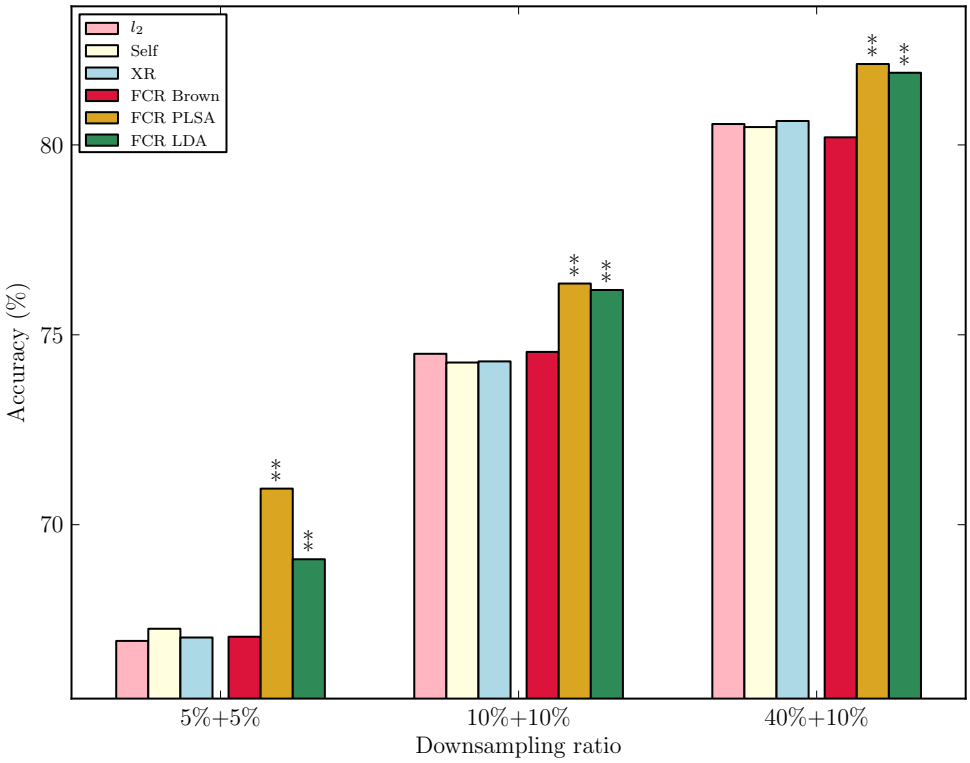


Figure 4.9: 20 Newsgroups topic classification results (FCR)

Clustering	Downsampling ratio		
	10%	20%	50%
Brown	10^{-10}	10^{-10}	10^{-9}
PLSA	100	100	1000
LDA	10^{-5}	0.0001	0.0001

Table 4.5: 20 Newsgroups FCR optimal hyper-parameters

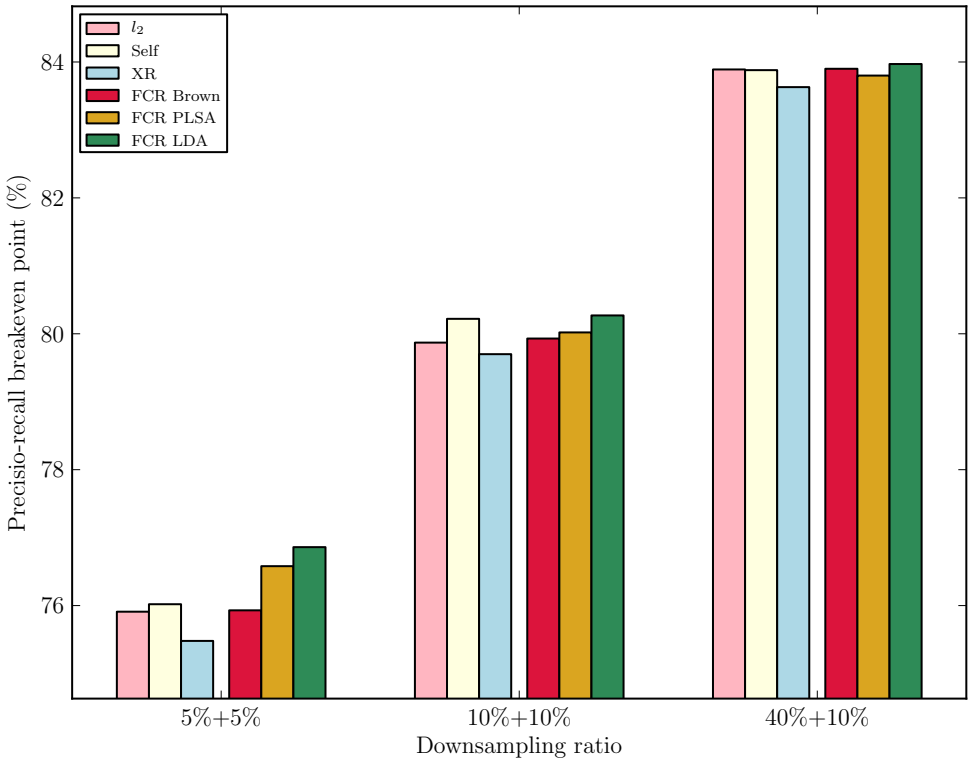


Figure 4.10: Reuters-21578 topic classification results (FCR)

Clustering	Downsampling ratio		
	10%	20%	50%
Brown	10^{-10}	10^{-10}	10^{-6}
PLSA	10^{-10}	10^{-10}	1
LDA	10^{-9}	10^{-9}	10^{-5}

Table 4.6: Reuters-21578 FCR optimal hyper-parameters

Brown-style word clustering is more targeted to word prediction, while PLSA and LDA are more targeted to topic modeling. Word prediction may not be very useful in the tasks

we investigated, but topic modeling is definitely helpful for topic classification. The trend of hyper-parameter change is similar to that of FAR (tables 4.4 – 4.6). In most cases, γ increases as overall downsampling ratio increases.

Although we claimed that FCR has potential computational advantage over FAR, no significant computation time difference is observed. This is because the $\mathbf{W}$ we learned is sparse, with the number of non-zero elements not being significantly greater than that of $\mathbf{M}$.

Between FAR and FCR, FAR obtains greater improvements on average than FCR does for the same task. In Reuters-21578 topic classification, FAR obtains significant gains when the overall downsampling ratio is 10%, whereas FCR does not improve the results significantly.

Despite both being topic classification tasks, different results are obtained at the same downsampling ratios for 20 Newsgroups and Reuters-21578 topic classification. Both FAR and FCR achieve more significant improvements for 20 Newsgroups than for Reuters-21578. We conjecture that the Reuters-21578 topic classification task suffers less from training data sparsity, because in our pilot study, we are able to obtain greater improvements from FAR and FCR at overall downsampling ratios at 1%, 2%, and 5%. But the accuracies at these downsampling ratios are too small to be useful, so we do not list them here. Furthermore, the test set of Reuters-21578 data set is the smallest among the three tasks in evaluation. Consequently, it is more difficult to obtain statistically significant improvements.

Lastly, we observe that FPWP provides faster convergence for unseen feature weights. Without FPWP, the update of unseen feature weights is much slower (figure 4.11). The impact of FPWP on seen feature weights is small. This can also be observed from the change of objective function value (figure 4.12), as the convergence of objective function value is similar for both cases. Note that the jumpiness of the curves is caused by the automatic adjustment of step size in L-BFGS. Even with FAR/FCR, seen feature weights still dominate the objective function, because only the seen feature weights can affect the loglikelihood term, and the optimally chosen γ is typically small.

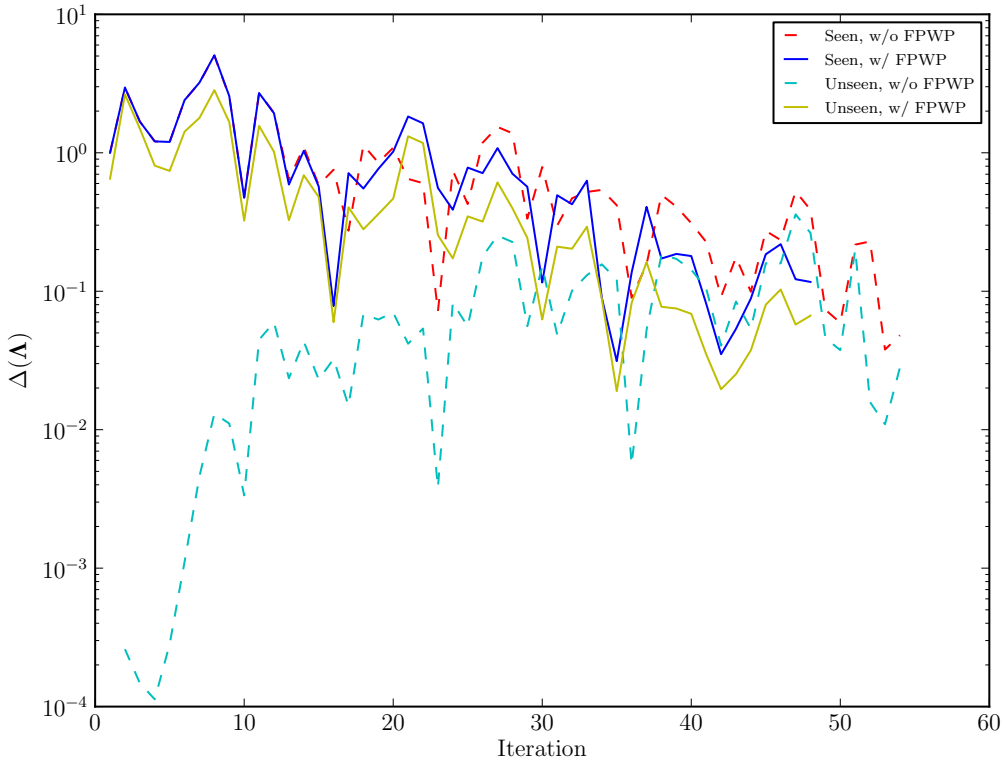


Figure 4.11: Feature weight change with vs. without FPWP in movie review classification. FAR with NCOOC is used. The jumpiness of the curves is caused by the automatic adjustment of step size in L-BFGS.

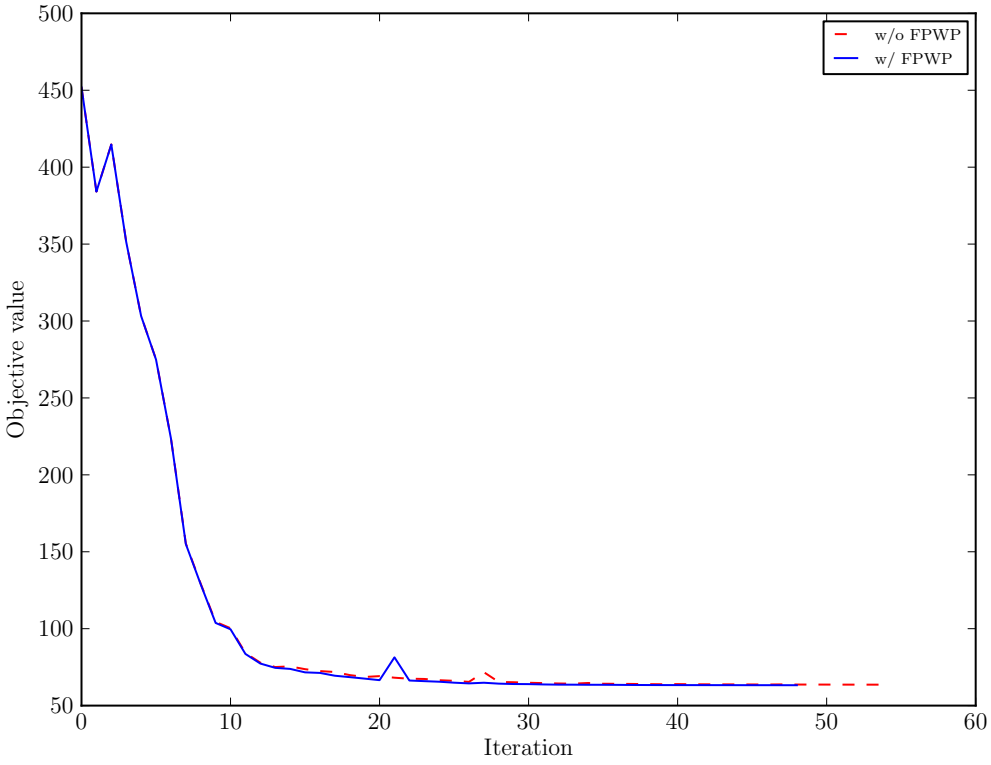


Figure 4.12: Training objective function value with vs. without FPWP in movie review classification. FAR with NCOOC is used.

4.7 Summary

In the section, we have proposed to learn feature relationships from unlabeled data and use this information to help supervised learning in two forms of regularization, including FAR and FCR. Feature affinity is learned based on multiple feature co-occurrence statistics, and both hard and soft feature clustering learning methods have been investigated. Although FAR models similarities between features directly, FCR is able to achieve more computational savings. FPWP has been proposed to speed up weight update for unseen features, and we have incorporated it with gradient-based MaxEnt training. We have evaluated the performance of these algorithms in three tasks, including movie review sentiment classi-

fication, 20 Newsgroups topic classification, and Reuters-21578 topic classification. Word features are used in these experiments, and both FAR and FCR have achieved statistically significant improvements over both supervised and semi-supervised baselines.

Chapter 5

MODELING HIGH-DIMENSIONAL FEATURE RELATIONSHIPS

In the previous chapter, we applied FAR and FCR to text classification problems where word features are commonly used. However, for many text classification problems, high-dimensional features, such as n-grams and phrase patterns, are shown to be useful in a supervised setting. Therefore, it is interesting to study if FAR and/or FCR can bring additional gains for high-dimensional features by leveraging unlabeled data. Moreover, unlike the tasks in the previous chapter, where we artificially created unlabeled data sets that were not much larger than the labeled training set, many real world tasks have abundant unlabeled data available to us. Hence, it is desirable to investigate methods of scaling up FAR and FCR for large data sets.

Not all methods introduced in the previous chapter about FAR are appropriate in this new scenario. For example, the methods for learning feature affinity are all based on feature co-occurrence. They need to be modified to accommodate a large number of features, as the size of a full co-occurrence matrix is $N \times N$, where N is the number of features and it can easily get as large as from hundreds of thousands to millions. Learning n-gram language models from large amounts of data has been extensively studied in recent decades [Seymore and Rosenfeld, 1996, Chen and Goodman, 1996, Goodman, 2001, Stolcke, 2002, Stolcke et al., 2011]. N-gram counts are collected from the data, possibly over multiple machines in parallel, and a compact language model can be generated by pruning the counts and using back-off strategies. Although the resulting language model may be small, full n-gram counts still need to be collected before any pruning. On the contrary, little work has been done on learning n-gram co-occurrence models from large data sets. Although Lin [2008] have studied word co-occurrence collection from large data sets using MapReduce [Dean and Ghemawat, 2008], n-gram features are much more high-dimensional than word features, the applicability of MapReduce to n-gram co-occurrence collection is unknown. If we use n-

grams as features, learning their co-occurrence can be much more resource-demanding than learning an n-gram language model. To see that, collecting the raw n-gram counts requires at most $O(NV)$ storage, where N is the number of distinct n-gram histories, and V is the vocabulary size. The maximum storage space for the co-occurrence counts for the same set of n-grams is, however, $O(N^2)$. Because $N \gg V$, co-occurrence counting is much more expensive. Typically, trigram features are used for many text classification tasks. Based on our analysis above, this is much more expensive than learning a 4-gram language model.

The FCR methods introduced previously are more scalable than FAR, as the size of feature clustering matrix is only $O(NK)$, where K is the number of clusters and can be controlled in size. We showed in section 4.2.2 that FCR is equivalent to FAR for a particular mixture co-occurrence definition that corresponds to a probabilistic soft clustering. This enables us to use a computationally more efficient FCR implementation, while still taking advantage of the superior FAR performance demonstrated in chapter 4. There are also efficient methods for learning mixture models, which we will explore in this chapter to learn feature co-occurrence.

Hofmann and Puzicha [1998] first used mixture models to model co-occurrence data, where pairs of samples are assumed to be generated from a mixture of distributions. PLSA was proposed by Hofmann a year later [Hofmann, 1999] to model document-word co-occurrence. Recently, big data processing is becoming a popular research area, and a lot of work has been devoted to development fast, distributed, and scalable implementations of PLSA. Hong et al. [2008] proposed a fast implementation of PLSA on a multi-core machine, where the co-occurrence matrix is divided into blocks and processed by different cores. Chen [2011] proposed a PLSA implementation based on a graphic processing unit (GPU). By using a 480-core GPU, a 33x speedup over a non-GPU single-core implementation was achieved. Both approaches above assume the existence of memory that is globally accessible to parallel cores, which is not widely available. Jin et al. [2011] proposed P^{2}LSA and P^{2}LSA+ as distributed PLSA implementations based on the Hadoop platform, an open-source implementation of MapReduce. Similarly, Li et al. [2012] proposed a parallel PLSA algorithm called PPLSA to accommodate large corpus collections in the MapReduce framework. Both approaches are based on MapReduce [Dean and Ghemawat, 2008], which can

utilize a cluster of machines. Our proposed algorithm is very similar to these approaches, but it is customized to model feature co-occurrence. We will show that parallel learning algorithms for PLSA can be easily modified to learn mixture models of feature co-occurrence.

In the following sections, we will discuss how high-dimensional feature co-occurrence can be modeled from large amounts of unlabeled data, including a direct co-occurrence model with MapReduce, and a mixture co-occurrence model. The direct co-occurrence model is more resource demanding. The mixture co-occurrence model is an approximation of the co-occurrence probabilities, but it is less costly to run. The mixture co-occurrence model introduced here is similar to PLSA in terms of the FCR regularization term, but it is more targeted to feature co-occurrence and thus is slightly different from PLSA in terms of learning the feature-cluster probabilities. We carry out experiments on tasks introduced in chapter 4, and alignment move and authority claim classification for Wikipedia discussions, using FCR with feature clustering derived from the mixture co-occurrence model.

5.1 Co-occurrence Models

5.1.1 Direct Co-occurrence Model

To model feature co-occurrence as feature affinity directly, we can compute all the feature co-occurrence statistics as described in the previous chapter based on the unlabeled data. The resulting co-occurrence can be used as the feature affinity matrix $\mathbf{W}$ for FAR. If the amount of unlabeled data is large and the feature dimensionality is high, the number of non-zero elements in the resulting co-occurrence matrix and thus $\mathbf{W}$ will generally be large. To reduce the training cost of FAR, we can further prune $\mathbf{W}$ to make it sparser, which saves storage space for $\mathbf{W}$ and speeds up regularizer evaluation and gradient computation. In order to do that, we still need to collect and save the full co-occurrence matrix.

When the size of unlabeled data is very large, parallel computing of co-occurrence statistics is desirable. MapReduce [Dean and Ghemawat, 2008] is a popular framework for distributed computing with lots of data, where we divide the unlabeled data into chunks and process each chunk independently over many machines. We do not use Hadoop because of its non-trivial deployment procedure. Rather, we implement a simple C++ program that

is able to perform basic MapReduce operations (e.g., map, partition, and reduce) on top of Condor. A major difference between this implementation and standard MapReduce is that we do not use a distributed file system. The data sets are kept on central file servers. It does not pose a problem for us, however, because the file size is on the order of gigabytes and can easily be handled by file servers. The same results can be obtained by using standard MapReduce implementations such as Hadoop. Here we store the unlabeled data in plain text organized in lines. We compute line-level feature co-occurrence unless otherwise noted.

Algorithm 6: Co-occurrence statistics collection with MapReduce

Input: Unlabeled text corpus $\mathcal{U}$

Output: Feature co-occurrence statistics

1 Collect preliminary feature statistics based on $\mathcal{U}$;

2 Partition $\mathcal{U}$ into m chunks of approximately equal size;

3 Dispatch multiple machines (mappers) to process these chunks, each mapper processing one chunk of data independently by executing algorithm 7. For each mapper, partition its results into r disjoint sets;

4 Dispatch multiple machines (reducers) to merge these partitioned results, the i-th reducer merging the i-th partition from all mappers' results by executing algorithm 8;

5 The merged results from the reducers are combined, optionally pruned, and saved to disk.

Algorithm 6 describes the MapReduce procedure for feature co-occurrence statistics collection. Because a single machine does not have the resources to process all the data in a speedy manner, the unlabeled text corpus $\mathcal{U}$ is divided evenly into m chunks, and each chunk is processed by a single machine. Before we do that, we may need to collect preliminary feature statistics from unlabeled data, depending on the particular co-occurrence statistics being used. For example, if we want to compute normalized co-occurrence probability, then we should collect the probability mass function of all features ahead of time. If we want to compute the Jaccard index, then we should collect the counts of samples in which each feature appears. These statistics serve as global information, and will be accessible to all

mappers and reducers.

Because each mapper only processes a chunk of the unlabeled data (algorithm 7), the time and memory requirements of each mapper are decreased compared to single machine approaches, and the amount of reduction is controlled by the number of chunks. The more chunks we divide $\mathcal{U}$ into, the less computation is needed for each mapper, but we also need to run mappers on more chunks so the overall CPU time may increase.

Algorithm 7: Mapper routine for algorithm 6

Input: Unlabeled text corpus $\mathcal{U}$

Output: A list of key-value pairs P containing partial co-occurrence statistics

```
1 Initialize empty hash table H, keys being feature pairs and values being integers;
2 foreach line in U do
3 |   Extract features for the current line;
4 |   foreach feature pair (f_1, f_2) do
5 |   |   Increment H[(f_1, f_2)] by one;
6 |   end
7 end
8 Convert the content of H into P.
```

As suggested in MapReduce, each mapper's results must be partitioned to be evenly distributed to r reducers. It is crucial because each reducer will only have to read and process a subset of feature pairs, decreasing reducer's CPU and memory requirements. The partitioning is accomplished by computing the hash values of the keys of the results, where the keys are feature pairs. Then each hash value's modulo of r is calculated, and it indicates the index of the reducer to which this result should be sent. A good hash function will ensure balanced traffic to each reducer, and we use the hash function implemented in the Boost C++ library. The reducer's complexity can be controlled by the number of partitions r. Each reducer maintains a hash table of feature pairs and co-occurrence statistics, and the co-occurrence statistic of each feature pair is accumulated as the mapper results are read in (algorithm 8).

Algorithm 8: Reducer routine for algorithm 6

Input: A list of key-value pairs P containing partial co-occurrence statistics

Output: A list of key-value pairs Q containing merged co-occurrence statistics

```
1 Initialize empty hash table H, keys being feature pairs and values being float
  numbers;
2 foreach key-value pair (k, v) in P do
3 |   Increment H[k] by v;
4 end
5 Convert the content of H into Q.
```

Although this algorithm utilizes multiple machines, it can still get quite expensive when the amount of unlabeled data is very large. Ignoring sparsity, the size of the co-occurrence matrix is the square of the number of features ($O(N^2)$). Therefore, the number of chunks m and partitions r, and thus the hardware resource, has to grow in second order as the feature dimensionality grows, which is often not practical. Its problem of scalability lies in the fact that, although the final $\mathbf{W}$ can be pruned to be very sparse, we need to collect and store the full $\mathbf{W}$ before we can start pruning, which can become very costly. We were not able to run this algorithm for the English Wikipedia discussion corpus (feature dimensionality is $N = 613,712$, including unigrams, bigrams, and trigrams) with $m = 197$ and $r = 30$ utilizing 50 machines, each having 8GB memory. Although it may be possible to run on more machines in a bigger cluster, we do not deem this algorithm to be very scalable due to the quadratic resource requirement.

5.1.2 Mixture Co-occurrence Model

Instead of keeping all co-occurrence statistics before pruning, we can resort to a less costly approximation of co-occurrence. We showed (section 4.2.2) that FCR is equivalent to FAR when feature co-occurrence is characterized using

$$p(f_1, f_2) = \sum_z p(f_1|z)p(f_2|z)p(z), \tag{5.1}$$

where f_1 and f_2 are two features, and z is a discrete latent variable taking on a predefined number of values (denoted Z). Note that $p(f_1|z)$ and $p(f_2|z)$ come from the same underlying distribution $p(f|z)$. The model parameters $p(f|z)$ and $p(z)$ can be learned using EM algorithm. This mixture co-occurrence model is in fact an approximated low-rank decomposition of the $N \times N$ feature co-occurrence matrix $\mathbf{C}$,

$$\mathbf{C} = \mathbf{Q}^T \mathbf{\Delta} \mathbf{Q}, \tag{5.2}$$

where $\mathbf{Q}$ is a $Z \times N$ matrix, and $\mathbf{\Delta}$ is a $Z \times Z$ diagonal matrix. The matrix elements satisfy

$$q_{zf} = p(f|z), \tag{5.3}$$

$$\delta_z = p(z), \tag{5.4}$$

and the feature clustering matrix is computed by

$$\mathbf{M} = \mathbf{Q}^T \mathbf{\Delta}. \tag{5.5}$$

Singular value decomposition (SVD, and here equivalent to eigenvalue decomposition as co-occurrence matrices are symmetric) can also achieve a similar low-rank decomposition, minimizing the Frobenius norm of the difference between the original co-occurrence matrix and the approximation. Non-negative matrix factorization (NMF) [Lee and Seung, 2000] is another related method, with additional non-negative matrix constraints. Compared to these methods, the mixture co-occurrence model seeks an approximation that maximizes the unlabeled data likelihood. Furthermore, it does not require loading the entire co-occurrence matrix into memory, which can be prohibitive. In each EM iteration, we go through the data once and only the sufficient statistics are stored in memory. Recently, Ding et al. [2008] showed that this type of mixture models and NMF with the I-divergence objective function optimize the same objective function, although they might reach different results because different optimization algorithms are used. The difference between PLSA and mixture co-occurrence models is that PLSA models feature-document co-occurrence, while mixture co-occurrence models characterize feature co-occurrence, which is directly related to the feature affinity measures introduced in section 4.4. Both can be used to generate feature

Algorithm 9: Mixture co-occurrence model parameter estimation with MapReduce

Input: Unlabeled text corpus $\mathcal{U}$

Output: Model parameters $p(f|z)$ and $p(z)$

Initialize parameters $p(f|z)$ and $p(z)$;
while *desired number of iterations is not reached* **do**
    Divide $\mathcal{U}$ into m chunks of approximately equal size;
    Dispatch multiple machines to process these chunks, each machine processing one chunk of data independently by executing algorithm 10;
    The mapper results are merged;
    Initialize empty hash table H, keys being feature-latent variable pair and values being float numbers;
    foreach *key-value pair* $((f, z), v)$ *in merged results* **do**
        Increment $H[(f, z)]$ by v;
    end
    foreach *feature in the feature set* **do**
        foreach z **do**
            $p(f|z) = \frac{H[(f,z)]}{\sum_f H[(f,z)]}$;
        end
    end
    foreach z **do**
        $p(z) = \frac{\sum_f H[(f,z)]}{\sum_{f,z} H[(f,z)]}$;
    end
end

Algorithm 10: Processing routine for algorithm 9

Input: Unlabeled text corpus $\mathcal{U}$, $p(f|z)$, $p(z)$

Output: A list of key-value pairs P containing partial feature-latent variable statistics

Initialize empty hash table H, keys being feature-latent variable pairs and values being float numbers;
foreach *line in* $\mathcal{U}$ **do**
    Extract features for the current line;
    foreach *feature pair* (f_1, f_2) *s.t.* $f_1 < f_2$ **do**
        foreach z **do**
            $q(z) = p(f_1|z)p(f_2|z)p(z)$;
        end
        foreach z **do**
            $p(z|f_1, f_2) = \frac{q(z)}{\sum_{z'=1}^{Z} q(z')}$;
            Increment $H[(f_1, z)]$ by $p(z|f_1, f_2)$;
            Increment $H[(f_2, z)]$ by $p(z|f_1, f_2)$;
        end
    end
end
Convert the content of H into P.

clustering, but the resulting feature clustering matrices will be different because the learning algorithms differ.

Similar to parallel learning of PLSA [Jin et al., 2011, Li et al., 2012], the model parameters of mixture co-occurrence models can be estimated using EM using multiple machines. Algorithm 9 illustrates EM estimation of the mixture co-occurrence model using parallel computing. In each EM iteration, the data chunks are sent to multiple machines for processing. The processing routine (algorithm 10) is the E-step, where the posterior probability mass function of the latent variable is computed for each feature co-occurrence pair, based on the current model parameters. The sufficient statistics, counts of feature-latent variable pairs, are saved as results. The M-step is carried out on a single local machine, where the merged sufficient statistics are used to update model parameters. Compared to algorithm 6, it only requires time and storage linear to N (i.e., $O(NZ)$). It scales much better as we only need to increase the number of machines linearly as the number of features grows, while keeping Z fixed. Algorithm 9 is a simplified instance of MapReduce, where only one reducer is employed. It is feasible because the sufficient statistics can usually fit in the memory of a single machine. Partitioning of mapper results is no longer necessary, as all results are sent to the same reducer. One might need multiple reducers in the case where feature dimensionality is very high.

We need to populate the matrices $\mathbf{W}$ and $\mathbf{M}$ to be able to use FAR and FCR, respectively. It is straightforward to apply the learned mixture co-occurrence model to build $\mathbf{M}$ in FCR, because we can treat the latent variable as the cluster index, and $p(f|z)p(z)$ can be used to construct the feature clustering matrix $\mathbf{M}$ as we did in the previous chapter. The same model can be applied to build $\mathbf{W}$ for FAR as well, if we use co-occurrence probability $p(f_1, f_2)$ to derive feature affinity, i.e., $w_{ij} = \sum_z p(f_i|z)p(f_j|z)p(z)$.

Since we have shown in the previous chapter that, using the above feature affinity and clustering, FAR and FCR are equivalent, they will yield the same results. Using FCR is cheaper than FAR, because if FAR is used, the full feature affinity matrix $\mathbf{W}$ has to be expanded from the mixture model. Therefore we only use FCR in our following experiments.

The equivalence also enables us to try feature affinity learning methods other than co-occurrence probability in the context of FCR. However, neither Jaccard index nor Dice

coefficient can be used, because they are not probabilistic methods and cannot be derived from the mixture co-occurrence model. Non-negative point-wise mutual information is not applicable, either, due to the non-decomposability of the logarithm. Only the normalized co-occurrence coefficient can be applied, if we assign

$$w_{ij} = \frac{\sum_z p(f_i|z)p(f_j|z)p(z)}{\sqrt{p(f_i)p(f_j)}}. \tag{5.6}$$

It can be further rewritten as

$$w_{ij} = \sum_z \frac{p(f_i|z)}{\sqrt{p(f_i)}} \frac{p(f_j|z)}{\sqrt{p(f_j)}} p(z). \tag{5.7}$$

If we let $q(f|z) = \frac{p(f|z)}{\sqrt{p(f)}}$, then $q(f|z)$ is no longer a valid conditional probability mass function since it is not guaranteed to sum to one. Nevertheless, we can compute $q(z) = \sum_f q(f|z)$, and use these quantities to renormalize $q(f|z)$[1],

$$w_{ij} = \sum_z \frac{p(f_i|z)}{\sqrt{p(f_i)}q(z)} \frac{p(f_j|z)}{\sqrt{p(f_j)}q(z)} p(z)q^2(z) \tag{5.8}$$

$$\propto \sum_z \tilde{p}(f_i|z)\tilde{p}(f_j|z)\tilde{p}(z), \tag{5.9}$$

where,

$$\tilde{p}(f|z) = \frac{p(f|z)}{\sqrt{p(f)}q(z)}, \tag{5.10}$$

$$\tilde{p}(z) = \frac{p(z)q^2(z)}{\sum_{z'} p(z')q^2(z')}. \tag{5.11}$$

They are valid probability mass functions. Therefore, we can use it in FCR by letting $m_{ic} = \tilde{p}(f_i|c)\tilde{p}(c)$.

5.2 Experiments on Word Features

We evaluate the performance of mixture co-occurrence models as defined in section 5.1.2, using word features. For the tasks introduced in chapter 4, it is feasible to compute the full

[1]The normalization of $q(f|z)$ is not only needed for stability of the hyper-parameter. It is also required for FAR and FCR to be equivalent. Although unnormalized probabilities are ready to be used for FAR, we still want to normalize them because our experiments are all done using FCR.

feature affinity matrix $\mathbf{W}$. The model introduced in equation (5.1) is a low-rank approximation of the co-occurrence-based $\mathbf{W}$, and the model defined in equation (5.9) is a low-rank approximation of the normalized co-occurrence-based $\mathbf{W}$. In this section, we compare the text classification performance when using the full-rank version of $\mathbf{W}$ vs. a reduced-rank approximation of $\mathbf{W}$.

5.2.1 Tasks and Experiment Paradigm

The three tasks introduced in section 4.6 are revisited here: movie review sentiment classification, 20 Newsgroups topic classification, and Reuters-21578 topic classification. The models introduced in equation (5.1) and equation (5.9) are employed with FCR, denoted by FCR COOC and FCR NCOOC, respectively. They will be compared with their full affinity matrix counterpart, i.e., FAR with co-occurrence probability (FAR COOC) and FAR with normalized co-occurrence probability (FAR NCOOC), respectively. FCR with PLSA (FCR PLSA) is also compared to, since it is a mixture model for topic. The results for FAR COOC, FAR NCOOC, and FCR PLSA are taken from chapter 4 without change.

5.2.2 Results

The experimental results for comparing mixture co-occurrence models with full feature affinity matrices are shown in figure 5.1 (movie review sentiment classification), figure 5.2 (20 Newsgroups topic classification), and figure 5.3 (Reuters-21578 topic classification). In the results shown in these figures, the number of clusters is $C = 10$. The statistical significance test results obtained using paired t-test are listed in table 5.1.

In most cases, we observe that the results with mixture co-occurrence models are worse than the results with full affinity matrices. That is not surprising because mixture co-occurrence models provide low-rank approximations to the full affinity matrices, and some affinity information may be lost during this approximation. This performance degradation is larger at low downsampling ratios than at higher downsampling ratios.

In movie review sentiment classification, the performance for most of the FCR mixture co-occurrence models falls between the results obtained with full affinity matrices and PLSA,

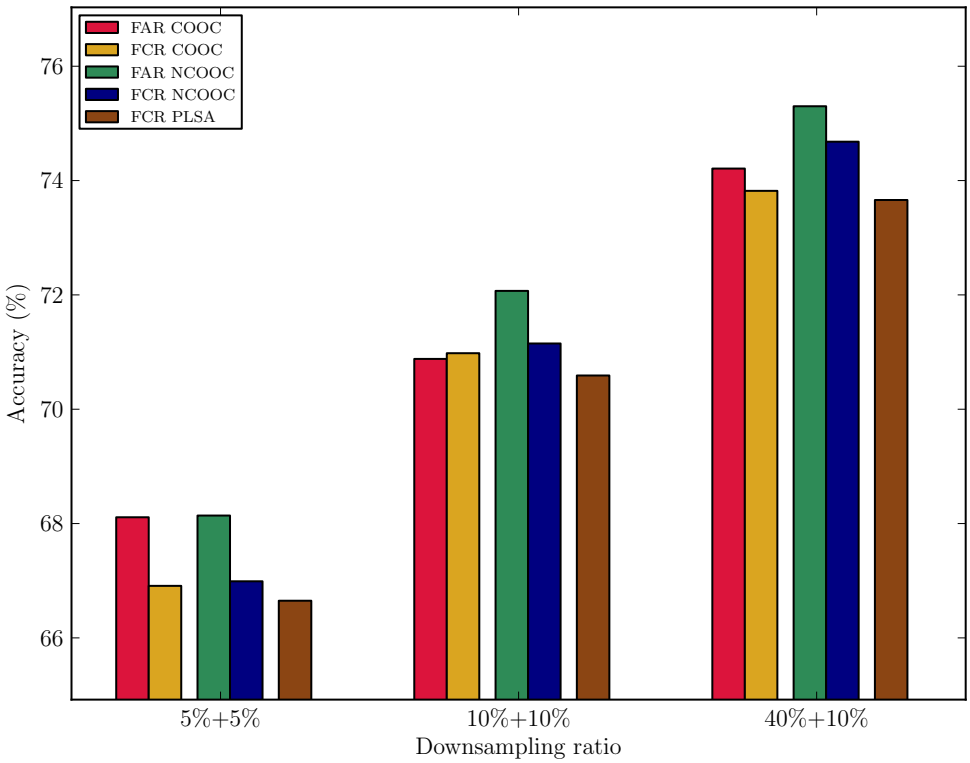


Figure 5.1: Movie review sentiment classification results with full **W** and mixture co-occurrence models

Approach		FCR COOC		FCR NCOOC	
Baseline		FAR COOC	FCR PLSA	FAR NCOOC	FCR PLSA
Downsampling ratio	5%+5%	**M-, N-**	**N-**	**M-, N-**, R-	**N-**
	10%+10%		**M+**	**M-**	M+, **N-**
	40%+10%	M-		**M-**, R-	**M+**

Table 5.1: List of tasks with significant difference ($p \leq 0.05$, $p \leq 0.01$ when in boldface) comparing FCR COOC/NCOOC to baselines (M: movie review sentiment classification, N: 20 Newsgroups topic classification, R: Reuters-21578 topic classification, +: significant improvement over baseline, -: significant degradation over baseline)

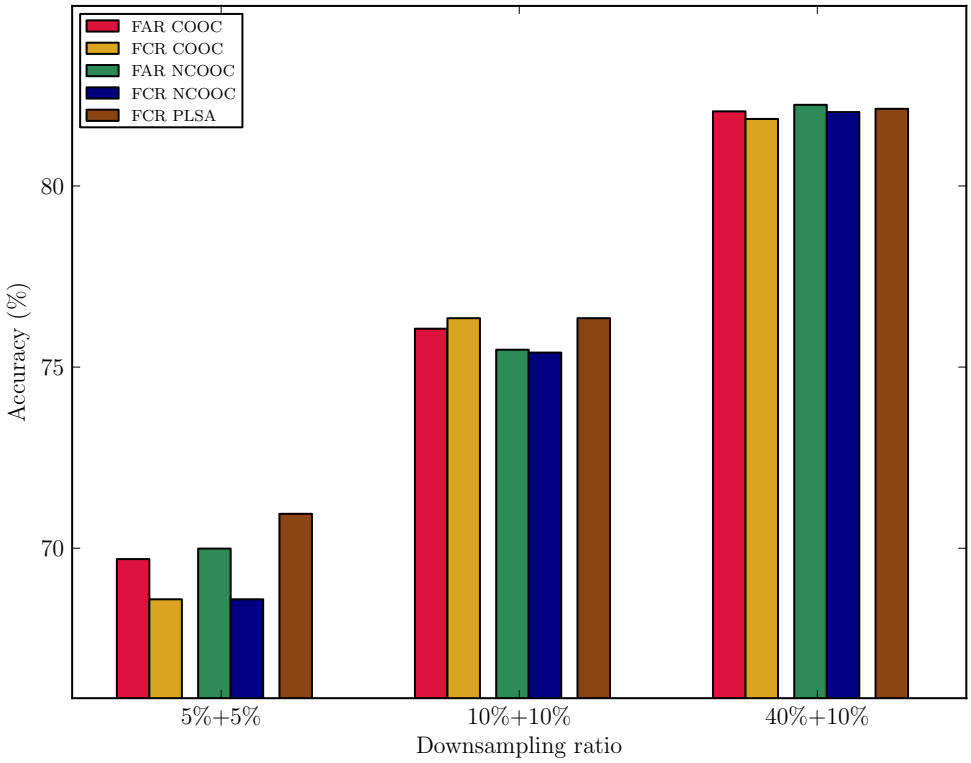


Figure 5.2: 20 Newsgroups topic classification results with full **W** and mixture co-occurrence models

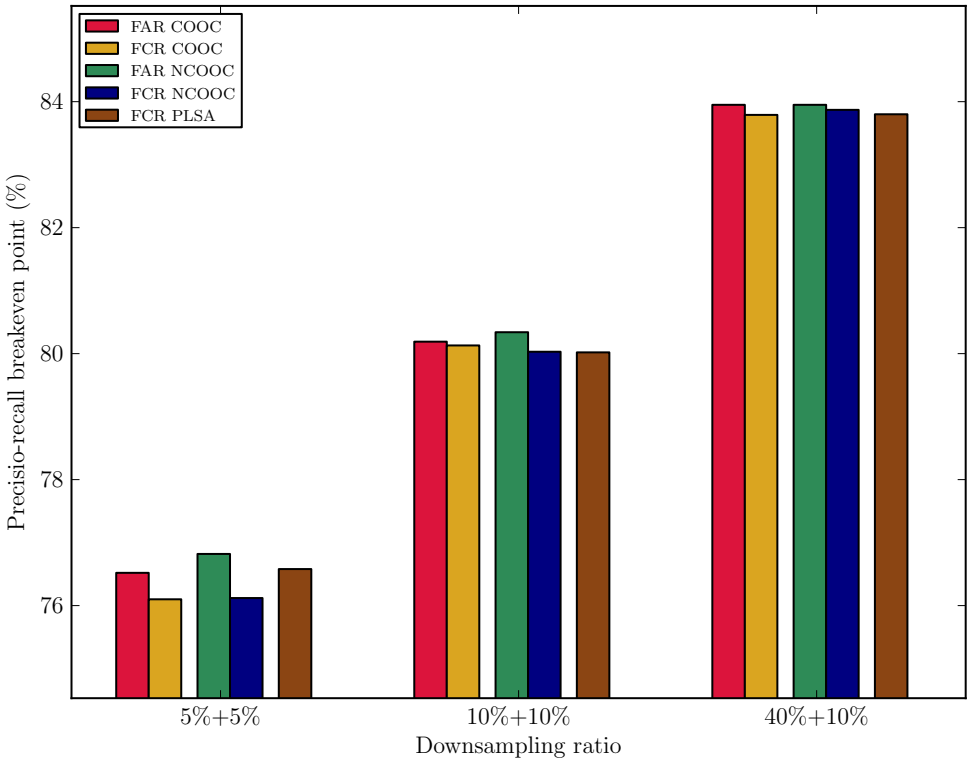


Figure 5.3: Reuters-21578 topic classification results with full **W** and mixture co-occurrence models

with full affinity matrices being the best performing and PLSA being the worst performing. This indicates that mixture co-occurrence models are better approximate models for sentence-level feature affinity associated with sentiment than PLSA is. In 20 Newsgroups and Reuters-21578 topic classification, mixture co-occurrence models do not perform so well as either full affinity matrices or PLSA at 10% overall downsampling ratios, which can be explained by the fact that PLSA is a topic model in nature. The performance difference between all models are quite small at 20% and 50% overall downsampling ratios.

Although only results with $C = 10$ are shown in the figures, we also experiment with $C = 100$. For FCR COOC and FCR NCOOC, changing C from 10 to 100 causes slight performance degradation for movie review sentiment classification, while performance gains are observed for 20 Newsgroups and Reuters-21578 topic classification. Especially, for 20 Newsgroups topic classification, using $C = 100$ is significantly better than not only FCR with $C = 10$ but the FAR counterparts. We conjecture that these different behaviors are caused by the differences in task and data, because the mixture co-occurrence model attempts to uncover latent feature clusters, and the optimal number of latent feature clusters are task-dependent. For FCR PLSA, changing C from 10 to 100 always leads to a performance degradation.

5.3 Experiments on High-Dimensional Features

In this section, we evaluate our algorithms on the Wikipedia discussion tasks that were introduced in chapter 3, which involve high-dimensional features that make mixture co-occurrence models desirable. Supervised learning methods were employed in that chapter, which left the large amount of unlabeled Wikipedia data unused. With FAR/FCR, it is possible to leverage those unlabeled data in a convenient way. It is also of particular interest because we know these tasks require high-dimensional features, which can be challenging to FAR/FCR due to the reasons mentioned above.

5.3.1 Unlabeled data

We downloaded the entire Wikipedia discussion data for English and Chinese dated at November 2012. Note that these data sets only include the current version of the discussions,

rather than all revisions of the discussions. The reason is that it greatly simplifies data processing, and we do not expect that the impact caused by all revisions will affect our tasks much, as different revisions are highly overlapped. Even with that, the English Wikipedia discussion data set is still very large, with about 20 million lines of text. Here the lines correspond to the paragraphs in formal text, but the size is usually much smaller due to the nature of online discussion forums. Each line may contain one or more sentences. As a comparison, the labeled English Wikipedia training data set is about 20 thousand automatically segmented sentences. Hence the unlabeled data is roughly more than 1000 times larger than the labeled data. The Chinese Wikipedia discussion data set, on the contrary, is much smaller, and contains about 0.2 million lines. But it is still much larger than the labeled data, which is about ten thousand automatically segmented sentences. We perform text cleaning and normalization that are almost identical to what we did for the labeled training data, but we do not segment lines or sentences. The line boundaries in the original posts are preserved.

5.3.2 Tasks and Experiment Paradigm

In our experiments, we investigate Wikipedia alignment move and authority claim classification for English and Chinese. As we did before (chapter 3), we treat the positive and negative alignment move classification as separate tasks, and, for authority claim classification, we carry out external and forum authority classification separately. All experiments are done in the form of five-fold cross-validation, and the results are reported in F-score, with the decision threshold tuned on development set.

We use the l_2 MaxEnt as the baseline.[2] Self training and XR could be valid baselines as well, but they too expensive to be feasible for large unlabeled data, because they need access to all the unlabeled data in each training iteration. This is another advantage that FAR/FCR brings, namely, we use abstracted knowledge about features learned from unlabeled data to help training, rather than using original unlabeled data directly. Contrast cases are the model defined in equation (5.1) with FCR (FCR COOC), and the model

[2]The baseline experiment also requires tuning of the decision threshold, therefore the comparison is fair even without development set fold-in.

defined in equation (5.9) with FCR (FCR NCOOC).

First, we use unigram, bigram, and trigram features in our experiments,[3] and we learn mixture co-occurrence models of these features on unlabeled data. To reduce computation and model size, we use only the features that appear in the labeled training and test data to construct the vocabulary. Furthermore, we remove all the features that only appear once in the unlabeled data, since the statistics of these features cannot be reliably estimated. The resulting vocabulary size for English Wikipedia discussion model is 613,712, and that for Chinese Wikipedia discussion model is 380,282. We use 50 machines to learn the mixture co-occurrence model, each having 1GB memory. The English Wikipedia discussion data is divided into 197 chunks, and the Chinese Wikipedia discussion data is divided into 21 chunks.

According to the experimental results in chapter 3, automatically learned phrase pattern features were able to achieve good results on these tasks. Therefore, we carry out an additional set of experiments and use the best results in chapter 3 as new baselines. The goal of these experiments is to verify weather the feature relationships learned from unlabeled data, although only about n-gram features, can achieve additional performance gains when phrase patterns are present. The set up of these experiments is largely the same as above, except for the new baselines. As both n-grams and phrase patterns are used as features, FCR applies to a subset of features, i.e., n-gram features.

5.3.3 Results

For the Wikipedia discussion-related tasks, the experimental results with n-gram features are shown in table 5.2 (English alignment move classification), table 5.3 (Chinese alignment move classification), table 5.4 (English authority claim classification), and table 5.5 (Chinese authority claim classification).

We test the statistical significance of the results using the approximate randomization test [Yeh, 2000]. Compared to the baseline results, if the $p \leq 0.01$, then we denote the corresponding result with two stars (**). If the $0.01 < p \leq 0.05$, then we denote the

[3]Phrase pattern features are not used because they are learned on a task-by-task basis, which means the mixture co-occurrence models have to be learned for each task and that is very costly.

corresponding result with one star (*). Otherwise, no star is shown.

Method	Positive	Negative
l_2	38.1%	38.8%
FCR COOC	**39.2%**	**39.3%****
FCR NCOOC	**39.2%****	39.3%

Table 5.2: English alignment move results (F-score) based on n-gram features

Method	Positive	Negative
l_2	26.7%	29.7%
FCR COOC	26.5%	**30.5%****
FCR NCOOC	**31.3%***	29.9%

Table 5.3: Chinese alignment move results (F-score) based on n-gram features

Method	External	Forum
l_2	49.5%	46.5%
FCR COOC	50.3%*	**46.8%**
FCR NCOOC	**50.8%****	45.8%

Table 5.4: English authority claim results (F-score) based on n-gram features

From the experimental results, we find that the proposed FCR methods with mixture co-occurrence model achieve significant gains in seven out of eight tasks (table 5.2 – 5.5). The only task where no significant gain is observed is English forum authority claim classification (table 5.4). Note that we learn feature affinity or clusterings from unlabeled data without any supervised information, which is not biased towards any particular task. However, the

Method	External	Forum
l_2	32.2%	32.3%
FCR COOC	32.4%	**38.3%***
FCR NCOOC	**34.2%****	38.1%*

Table 5.5: Chinese authority claim results (F-score) based on n-gram features

co-occurrence model may be biased. For example, it is possible that English forum authority claim cues do not co-occur frequently. Therefore, the feature affinity and clustering learned from mixture co-occurrence models may help each task differently.

FCR COOC achieves significant gains in four out of eight tasks, while FCR NCOOC achieves significant gains in five out of eight tasks (table 5.2 – 5.5). FCR COOC performs better than FCR NCOOC does in positive alignment move classification and external authority claim classification. On the other hand, FCR NCOOC does better in negative alignment move classification and forum authority claim classification. This observation seems consistent for both languages. The difference between COOC and NCOOC is the weighting of co-occurrence counts. The experimental results show that the downweighting of frequent features helps in negative alignment move and forum authority claim classification.

The experimental results with n-gram and phrase pattern features are shown in table 5.6 (English alignment move classification), table 5.7 (Chinese alignment move classification), table 5.8 (English authority claim classification), and table 5.9 (Chinese authority claim classification).

When both n-gram and phrase pattern features are used, we are able achieve significant improvements over a stronger baseline in three of the four English tasks (table 5.6 and 5.8). All significant improvements are obtained by FCR NCOOC. Unfortunately, we are not able achieve significant gains for any Chinese tasks (table 5.7 and 5.7). Instead, we see some over-training. This may be explained by the fact that the unlabeled Chinese Wikipedia discussion data set we learn the feature clustering from is relatively small (0.2 million lines, vs. 20 million lines for English) and so it is more difficult to obtain further gains on top of

Method	Positive	Negative
l_2	40.8%	40.5%
FCR COOC	42.2%*	40.4%
FCR NCOOC	**43.1%****	**41.5%****

Table 5.6: English alignment move results (F-score) based on n-gram and phrase pattern features

Method	Positive	Negative
l_2	**33.9%**	31.5%
FCR COOC	30.5%	31.4%
FCR NCOOC	30.4%	**32.3%**

Table 5.7: Chinese alignment move results (F-score) based on n-gram and phrase pattern features

phrase pattern features.

5.4 Summary

In this section, we have described challenges that we face when we use FAR/FCR with high-dimensional features on large amounts of unlabeled data. We have shown that direct collection of high-dimensional feature co-occurrence counts can be expensive, even with MapReduce. A more tractable model based on PLSA, the mixture co-occurrence model, has been introduced in the FCR framework, and it is shown to be equivalent to FAR with a particular feature affinity measure. It can be learned efficiently using the EM algorithm on multiple machines. Experiments on Wikipedia alignment move and authority claim classification have been carried out, where the feature affinity and clustering are learned from large unlabeled Wikipedia discussion data. Significant improvements over supervised training on alignment move and authority claim classification have been achieved for n-gram features, although there is no single co-occurrence function that is consistently best. In

Method	External	Forum
l_2	48.9%	**46.8%**
FCR COOC	48.6%	46.6%
FCR NCOOC	**49.8%****	46.1%

Table 5.8: English authority claim results (F-score) based on n-gram and phrase pattern features

Method	External	Forum
l_2	**34.3%**	**40.3%**
FCR COOC	31.8%	37.4%
FCR NCOOC	30.9%	39.9%

Table 5.9: Chinese authority claim results (F-score) based on n-gram and phrase pattern features

addition, performance improvement over a strong baseline using phrase features is achieved for the English tasks.

Chapter 6

CONCLUSION

6.1 Contributions

The main contribution of this thesis can be summarized as two aspects of feature learning:

Learning phrase patterns as new features We have proposed an approach to efficiently learn phrase pattern features as new features for supervised text classification.

Learning and utilizing feature relationships We have proposed an approach to learn feature relationships from unlabeled data, which is then used in semi-supervised text classification.

We have evaluated the proposed algorithms in a variety of text classification tasks, including topic, sentiment, speaker role, alignment move, and authority claim classification. Significant improvements have been observed. The state of the art on speaker role, alignment move, and authority claim classification has been advanced.

6.1.1 Learning Phrase Patterns as New Features

Phrase pattern features are generalizations of n-gram features, and they are less rigid and have more discriminative power for many text classification problems. However, selecting a good set of phrase pattern features from the exponentially many phrase pattern candidates in text is a challenging problem.

In chapter 3, we have proposed an extended *PrefixSpan* algorithm to learn phrase pattern features from labeled text data. *PrefixSpan* is a well-known algorithm to efficiently extract frequent sequential patterns from unlabeled data. Our extension to *PrefixSpan* converts it to a supervised method, allowing the use of discriminative pattern selection criteria other than frequency. This extension is fundamental to this thesis, because good, discriminative phrase

patterns are not necessarily frequent, and label information is indispensable to learning these features. We use mutual information as a criterion to select discriminative phrase patterns, and it integrates nicely with the prefix-growing approach used in *PrefixSpan*.

When labeled training data is limited, the use of word classes is more desirable, because it is easier to get more reliable count estimation for word classes than for words, and word classes are more likely to re-occur once they are seen. Extended *PrefixSpan* also enables flexible use of word classes in phrase pattern features. A certain slot in a phrase pattern may be occupied by either words, word classes, or a combination of both. The algorithm searches for the combination that has the most discriminative power. Several choice of word classes have been discussed. Not surprisingly, there is no one type of word class that is found to be the best word class for all tasks in the experiments, but we have found that domain-dependent data-driven word classes and manually-compiled word classes tend to work better.

The tasks investigated in our experiments are speaker role classification for broadcast conversations, Wikipedia discussion alignment move classification, and Wikipedia discussion authority claim classification. All tasks involve both English and Chinese text, and speaker role classification also involves text from ASR output. Significant improvements have been observed by adding phrase pattern features into the feature set.

6.1.2 Learning and Utilizing Feature Relationships

When there is much more unlabeled data than labeled data, we see many features that only appear in unlabeled data but not labeled data. A conventional supervised classifier ignores these unseen features, or, equivalently, it assigns zero weights to these features.

In chapter 4, we have proposed feature affinity regularization and feature cluster regularization, which directly address this issue. By designing regularizers that favor similar weights for similar features, we are able to push the weights of similar features closer. We have shown these regularizers are convex and can be easily integrated into gradient-based MaxEnt training. These regularizers not only regularize seen feature weights, they also link seen and unseen feature weights, thereby propagating non-zero weights to unseen features.

Our proposed feature cluster regularization allows the use of both hard and soft feature clusterings, which, to the best of our knowledge, is the first work that utilizes soft feature clustering information to improve text classification.

Various ways of learning feature affinity and clustering from unlabeled text data have been introduced. These approaches have been designed to be general over different tasks, and they do not require specific application-dependent knowledge. We have shown that, under certain feature affinity and clustering assumptions, feature affinity regularization and feature clustering regularization are in fact equivalent.

We have proposed fixed point weight propagation, an improvement to pure gradient-based update of unseen feature weights. This algorithm works in conjunction with conventional gradient-based MaxEnt training, and can greatly speed up the convergence of unseen feature weights. We have also found that it works best if we only use l_2 regularization on seen feature weights.

Experiments have been carried out on a few topic classification tasks where word features are used, including movie review sentiment classification, 20 Newsgroups topic classification, and Reuters-21578 topic classification. We have observed significant improvements by using artificially downsampled training data. Among different feature affinities and clusterings, normalized co-occurrence probability and PLSA-based word clustering are consistently well-performing.

The word features used in chapter 4 are not very high-dimensional relative to the n-gram and phrase pattern features used in many tasks, and therefore represent a limited test of our algorithms. Chapter 5 extends that by investigating and addressing the practical issues of feature regularization for high-dimensional features. A mixture co-occurrence model has been proposed to learn feature affinity and clustering from large data sets using cluster of machines. Evaluating on movie review sentiment classification, 20 Newsgroups topic classification, and Reuters-21578 topic classification, this model has achieved slightly worse performance than FAR with (normalized) co-occurrence probability measures. It also has outperformed FCR with PLSA on movie review sentiment classification, although the reverse has been observed on 20 Newsgroups and Reuters-21578 topic classification. Because of easy access to the large amount of unlabeled Wikipedia data, we have evaluated this method

in the Wikipedia-related tasks introduced in chapter 3. Significant improvements have been achieved in most tasks by using feature cluster regularization with high-dimensional n-gram features. Additional gains are achieved by combining semi-supervised feature cluster regularization with the phrase features previously obtained through supervised learning.

6.2 Future Directions

There are various ways in which this work may be extended. We envision a few future directions based on this thesis, including semi-supervised phrase pattern learning, online learning of feature relationships, FAR/FCR for phrase pattern features, and the incorporation of dependency information.

For newly introduced text classification tasks, there will always be a shortage of labeled training data. Learning phrase pattern features with word classes using extended *PrefixSpan* can alleviate the data sparsity problem to some extent, but it does not fully make use of unlabeled data. To overcome this problem, semi-supervised phrase pattern learning is desirable. A straightforward approach would be modifying the phrase pattern search criterion to not only include labeled data but also unlabeled data. We tried a criterion which contains two requirements: the mutual information between a phrase pattern and labels computed on labeled data must be greater than a threshold, and the frequency of a phrase pattern on unlabeled data must be greater than a threshold. However, no significant improvement has been observed. The reason may be that, due to computational reasons, we can only use a small unlabeled data set, which does not contain enough information about phrase patterns. In order to do this over a much larger unlabeled data set, such as the entire Wikipedia discussion data used in chapter 5, a parallelized extended *PrefixSpan* is desirable. It is a challenging problem, and Sutou et al. [2003] proposed an approach of parallelized *PrefixSpan*, which uses message passing interface (MPI) to achieve dynamic load balancing.

The parallelized mixture co-occurrence model training proposed in chapter 5 greatly speeds up the learning of feature affinity and clustering from a large text data set. But with ever increasing text data on the web, it is useful to develop online learning algorithms for feature relationships. Algorithm 9 is a batch algorithm. When new text data comes in,

we need to add the new data to the existing data, and retrain the entire model. This is a requirement of the EM algorithm, because, in each iteration, the whole data set must be available. Online algorithms update the model incrementally. Therefore, we do not need to revisit existing data when updating the model, which is much more efficient. Several implementations of online EM have been proposed in recent literature [Cappé and Moulines, 2011, Liang and Klein, 2009], which may be applied to solve this problem. To achieve maximum efficiency, an online and parallelized EM algorithm for the mixture co-occurrence model is preferred. Furthermore, in order to address an evolving mixture co-occurrence model in some applications (due to, e.g., new topics or products), algorithms may dynamically modify the number of mixture components and update the regularization.

Chapter 5 has shown significant text classification improvements by adding phrase pattern features and applying FCR, where FCR is applied to n-gram features only. In theory, it is possible to explore FCR for all features including both n-gram and phrase pattern features. But there is a large cost associated with the implementation. Unlike n-gram features, which can be extracted on the fly, the extraction of phrase pattern features requires the text to be tagged with word classes. Word classes are usually tagged ahead of time, which may add significant computation and storage overhead to the process if a very large amount of unlabeled data is involved. Efficient word class tagging algorithms may be helpful in this case.

It is possible to extend this work to incorporate dependency information. One approach is to add dependency-based word classes in phrase patterns. Dependency-based word classes are derived from dependency parses. For example, the head word or the word class of the head corresponding to the current word can be used as a word class of the current word, after dependency parses are obtained. Another approach to incorporate dependency information in this work is to learn feature affinity and clustering information for features including dependency features. We can then learn similarities between dependency features and other features, and between dependency features themselves. With that information, FAR/FCR can be applied.

BIBLIOGRAPHY

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of International Conference on Very Large Data Bases*, pages 487–499.

Allison, B. (2008). Sentiment detection using lexically-based classifiers. In *Text, Speech and Dialogue*, volume 5246 of *Lecture Notes in Computer Science*, chapter 5, pages 21–28.

Amine, B. M. and Mimoun, M. (2007). Wordnet based cross-language text categorization. In *IEEE/ACS International Conference on Computer Systems and Applications*, pages 848–855.

Andrew, G. and Gao, J. (2007). Scalable training of l_1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40.

Baker, L. D. and McCallum, A. K. (1998). Distributional clustering of words for text classification. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103.

Barbosa, L. and Feng, J. (2010). Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of COLING*, pages 36–44, Beijing, China.

Barzilay, R., Collins, M., Hirschberg, J., and Whittaker, S. (2000). The rules behind roles: Identifying speaker role in radio broadcasts. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 679–684. AAAI Press.

Bekkerman, R., Yaniv, R. E., Tishby, N., and Winter, Y. (2003). Distributional word clusters vs. words for text categorization. *J. Mach. Learn. Res.*, 3:1183–1208.

Bender, E. M., Morgan, J. T., Oxley, M., Zachry, M., Hutchinson, B., Marin, A., Zhang, B., and Ostendorf, M. (2011). Annotating social acts: Authority claims and alignment

moves in wikipedia talk pages. In *Proceedings of Workshop on Language in Social Media*, pages 48–57.

Berger, A. L., Della Pietra, V. J., and Della Pietra, S. A. (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(4-5):993–1022.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*, pages 120–128.

Bloehdorn, S. and Hotho, A. (2004). Boosting for text classification with semantic features. In *WebKDD'04 Proceedings of the 6th international conference on Knowledge Discovery on the Web: advances in Web Mining and Web Usage Analysis*, pages 149–166.

Bloehdorn, S. and Moschitti, R. (2007). Combined syntactic and semantic kernels for text classification. In *Advances in Information Retrieval - Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007)*, volume 4425, pages 307–318.

Brown, P. F., deSouza, P. V., Mercer, R. L., Della Pietra, V. J., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208.

Cai, L. and Hofmann, T. (2003). Text categorization by boosting automatically extracted concepts. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, volume 5, pages 182–189.

Cappé, O. and Moulines, E. (2011). Online EM algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613.

Chen, H. (2011). Parallel implementations of probabilistic latent semantic analysis on graphic processing units. Master's thesis, University of Illinois at Urbana-Champaign.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. John Wiley & Sons. Inc, 2 edition.

Dagan, I., Lee, L., and Pereira, F. C. N. (1999). Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1):43–69.

Davidov, D. and Rappoport, A. (2006). Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *COLING-ACL*, pages 297–304.

Davidov, D., Tsur, O., and Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of CoNLL*, pages 107–116.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dhillon, I. S., Mallela, S., and Kumar, R. (2002). Enhanced word clustering for hierarchical text classification. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 191–200.

Dhillon, I. S., Mallela, S., and Kumar, R. (2003). A divisive information theoretic feature clustering algorithm for text classification. *J. Mach. Learn. Res.*, 3:1265–1287.

Ding, C., Li, T., and Peng, W. (2008). On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927.

Dong, G. and Pei, J. (2007). *Sequence data mining*. Springer.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition.

Elberrichi, Z. (2006). Using wordnet for text categorization. In *International Arab Journal of Information Technology*, volume 5, pages 3–37.

Feldman, S., Marin, M. A., Ostendorf, M., and Gupta, M. R. (2009). Part-of-speech histograms for genre classification of text. pages 4781–4784.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163.

Fukumoto, F. and Suzuki, Y. (2001). Learning lexical representation for text categorization. In *NAACL*, pages 232–240.

Galley, M., McKeown, K., Hirschberg, J., and Shriberg, E. (2004). Identifying agreement and disagreement in conversational speech: use of Bayesian networks to model pragmatic dependencies. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 669–676.

Gamon, M. (2005). Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of COLING*, pages 841–847.

Gilbert, E., Bergstrom, T., and Karahalios, K. (2009). Blogs are echo chambers. In *Proceedings of HICSS*, pages 1–10.

Goodman, J. (2004). Exponential priors for maximum entropy models. In *Proceedings of HLT-NAACL*, pages 305–312.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Proceedings of NIPS*, pages 529–536.

Guyon, I. and Elisseeff, A. (2006). An introduction to feature extraction. In *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 1–25. Springer Berlin Heidelberg.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer, 2nd ed. 2009. corr. 3rd printing 5th printing. edition.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz.

Hillard, D., Ostendorf, M., and Shriberg, E. (2003). Detection of agreement vs. disagreement in meetings: training with unlabeled data. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 34–36.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554.

Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in boltzmann machines. *Parallel Distributed Processing*, pages 282–317.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI'99*, pages 289–296.

Hofmann, T. and Puzicha, J. (1998). Statistical models for co-occurrence data. Technical report, Massachusetts Institute of Technology.

Hong, C., Chen, W., Zheng, W., Shan, J., Chen, Y., and Zhang, Y. (2008). Parallelization and characterization of probabilistic latent semantic analysis. In *Parallel Processing, 2008. ICPP. 37th International Conference on*, pages 628–635.

Hotho, A., Staab, S., and Stumme, G. (2003). Wordnet improves Text Document Clustering. In *In Proceedings of of the SIGIR 2003 Semantic Web Workshop*, pages 541–544.

Hutchinson, B., Zhang, B., and Ostendorf, M. (2010). Unsupervised broadcast conversation speaker role labeling. In *Proceedings of ICASSP*, pages 5322–5325.

Hwang, M.-Y., Peng, G., Wang, W., Faria, A., Heidel, A., and Ostendorf, M. (2007). Building a highly accurate mandarin speech recognizer. In *Proceedings of ASRU*, pages 490–495.

Ifrim, G., Bakir, G., and Weikum, G. (2008). Fast logistic regression for text categorization with variable-length n-grams. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 354–362.

Jaillet, S., Laurent, A., and Teisseire, M. (2006). Sequential patterns for text categorization. *Intell. Data Anal.*, 10(3):199–214.

Jalam, R. and Teytaud, O. (2001). Kernel-based text categorisation. In *Proceedings of International Joint Conference on Neural Networks*, pages 1891–1896.

Jensen, F. (1996). *An Introduction To Bayesian Networks*. CRC Press, 1 edition.

Ji, X., Bailey, J., and Dong, G. (2007). Mining minimal distinguishing subsequence patterns with gap constraints. *Knowledge and Information Systems*, 11(3):259–286.

Jin, Y., Gao, Y., Shi, Y., Shang, L., Wang, R., and Yang, Y. (2011). P^2LSA and P^2LSA+: Two paralleled probabilistic latent semantic analysis algorithms based on the MapReduce model. In *Intelligent Data Engineering and Automated Learning - IDEAL 2011*, volume 6936 of *Lecture Notes in Computer Science*, pages 385–393.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, Universität Dortmund, LS VIII-Report.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, chapter 19, pages 137–142.

Kim, S. M. and Hovy, E. (2006). Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 483–490.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324.

Kudo, T. and Matsumoto, Y. (2004). A boosting algorithm for classification of semi-structured text. In *Proceedings of EMNLP 2004*, pages 301–308.

Lee, D. D. and Seung, S. H. (2000). Algorithms for non-negative matrix factorization. In *NIPS*, volume 13, pages 556–562.

Lei, X., Wu, W., Wang, W., Mandal, A., and Stolcke, A. (2009). Development of the 2008 SRI Mandarin speech-to-text system for broadcast news and conversations. In *Proceedings of Interspeech*, pages 2099–2102.

Leslie, C., Eskin, E., and Stafford, W. (2002). The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 564–575.

Lewis, D. (1992). Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics.

Lewis, D. (1999). The reuters-21578 test collection. `http://www.daviddlewis.com/resources/testcollections/reuters21578`.

Lewis, D. and Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93.

Li, N., Zhuang, F., He, Q., and Shi, Z. (2012). PPLSA: Parallel probabilistic latent semantic analysis based on MapReduce intelligent information processing VI. volume 385 of *IFIP Advances in Information and Communication Technology*, pages 40–49.

Liang, P. and Klein, D. (2009). Online EM for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 611–619.

Lin, J. (2008). Scalable language processing algorithms for the masses: a case study in computing word co-occurrence matrices with MapReduce. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 419–428.

Liu, C.-L. (2005). Some theoretical properties of mutual information for student assessments in intelligent tutoring systems. *Foundations of Intelligent Systems*, 3488:77–93.

Liu, H. and Motoda, H., editors (1998). *Feature Extraction, Construction and Selection*. Springer US.

Liu, T. (2010). A novel text classification approach based on deep belief network. In *Proceedings of the 17th international conference on Neural information processing: theory and algorithms - Volume Part I*, ICONIP'10, pages 314–321. Springer-Verlag.

Liu, Y. (2006). Initial study on automatic identification of speaker role in broadcast news speech. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 81–84.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). The string kernel – text classification using string kernels. *Journal of Machine Learning Research*, (2):419–444.

Lui, A. K., Li, S. C., and Choy, S. O. (2007). An evaluation of automatic text categorization in online discussion analysis. In *Seventh IEEE International Conference on Advanced Learning Technologies*, pages 205–209.

Maas, A. and Ng, A. (2010). A probabilistic model for semantic word vectors. In *Deep Learning and Unsupervised Feature Learning Workshop NIPS 2010*, pages 1–9.

Mann, G. S. and McCallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of ICML*, pages 593–600.

Mann, G. S. and McCallum, A. (2010). Generalized expectation criteria for semi-supervised learning with weakly labeled data. *J. Mach. Learn. Res.*, 11:955–984.

Manning, C. D. and Schuetze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, 1 edition.

Marin, A., Ostendorf, M., Zhang, B., Morgan, J. T., Oxley, M., Zachry, M., and Bender, E. M. (2010). Detecting authority bids in online discussions. In *Proceedings of SLT*, pages 49–54.

Marin, A., Zhang, B., and Ostendorf, M. (2011). Detecting forum authority claims in online discussions. In *Proceedings of Workshop on Language in Social Media*, pages 48–57.

Martineau, J. and Finin, T. (2009). Delta TFIDF: An improved feature space for sentiment analysis. In *ICWSM*.

McCallum, A. and Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In *EACL '03 Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 307–314.

McCallum, A. K. (2002). MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244.

Morgan, J. and Oxley, M. (2010). Authority Annotation Guidelines. Technical report, University of Washington.

Morgan, J. T., Bender, E. M., Oxley, M., Zhu, L., Gracheva, V., and Zachry, M. (2013). Are we there yet?: The development of a corpus annotated for social acts in multilingual online discourse. *Dialogue & Discourse (Special Issue), under final revision for "Beyond semantics: the challenges of annotating pragmatic and discourse phenomena."*.

Ng, A. Y. (2004). Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 78–85. ACM.

Ng, V., Dasgupta, S., and Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 611–618.

Nigam, K., Mccallum, A., and Mitchell, T. (2006). Semi-supervised text classification using EM. In *Semi-Supervised Learning*, pages 33–56. MIT Press.

Okanohara, D. and Tsujii, J. (2009). Text categorization with all substring features. In *Proceedings of SDM*, pages 838–846.

Oxley, M., Morgan, J. T., Zachry, M., and Hutchinson, B. (2010). "what i know is...": establishing credibility on wikipedia talk pages. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, pages 271–278.

Pang, B. and Lee, L. (2005). Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-c. (2001).

PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of International Conference on Data Engineering*, pages 215–224.

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238.

Pennebaker, J., Chung, C., Ireland, M., Gonzales, A., and Booth, R. (2007). *The development and psychometric properties of LIWC2007.*

Pennebaker, J., Francis, M., and Booth, R. (2001). *Linguistic Inquiry and Word Count: LIWC2001.* Erlbaum Publishers.

Rennie, J. (2008). The 20 newsgroups data set. `http://people.csail.mit.edu/jrennie/20Newsgroups`.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Riloff, E. and Wiebe, J. (2003). Learning Extraction Patterns for Subjective Expressions. In *EMNLP '03 Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112.

Rockafellar, R. T. (1997). *Convex analysis.* Princeton University Press.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Sandler, T., Talukdar, P. P., Ungar, L. H., and Blitzer, J. (2008). Regularized learning with networks of features. In *Proceedings of NIPS*, pages 1401–1408.

Santini, M. (2004). A shallow approach to syntactic feature extraction for genre classification. In *7th Annual CLUK Research Colloquium.*

Schapire, R. E. and Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2):135–168.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 1st edition.

Scott, S. and Matwin, S. (1998). Text classification using WordNet hypernyms. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference, Association for Computational Linguistics*, pages 45–52.

Scott, S. and Matwin, S. (1999). Feature engineering for text classification. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 379–388.

Seymore, K. and Rosenfeld, R. (1996). Scalable backoff language models. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 1, pages 232–235.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge Univ Pr.

Socher, R., Manning, C., and Ng, A. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*.

Stolcke, A. (2002). SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.

Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). SRILM at sixteen: Update and outlook. In *Proceedings IEEE Automatic Speech Recognition and Understanding Workshop*.

Subramanya, A., Petrov, S., and Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of EMNLP*, pages 167–176.

Sun, G., Liu, X., Cong, G., Zhou, M., Xiong, Z., Lee, J., and Lin, C. Y. (2007). Detecting erroneous sentences using automatically mined sequential patterns. In *Proceedings of ACL*, pages 81–88.

Sutou, T., Tamura, K., Mori, Y., and Kitakami, H. (2003). Design and implementation of parallel modified PrefixSpan method. In *High Performance Computing*, volume 2858 of *Lecture Notes in Computer Science*, pages 412–422.

Terra, E. and Clarke, C. L. A. (2003). Frequency estimates for statistical word similarity measures. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 165–172.

Thomas, M., Pang, B., and Lee, L. (2006). Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *In Proceedings of EMNLP*, pages 327–335.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*, pages 173–180.

Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*, pages 63–70.

Tsur, O., Davidov, D., and Rappoport, A. (2010). ICWSM – a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proceedings of AAAI*.

Vinciarelli, A. (2007). Speakers role recognition in multiparty audio recordings using social network analysis and duration distribution modeling. *IEEE Transactions on Multimedia*, 9(6):1215–1226.

Wang, M., He, Y., and Jiang, M. (2010). Text categorization of Enron email corpus based on information bottleneck and maximal entropy. pages 2472–2475.

Wang, W. and Yang, J. (2005). *Mining Sequential Patterns from Large Data Sets*. Springer.

Wiebe, J. and Riloff, E. (2005). Creating subjective and objective sentence classifiers from unannotated texts. *Computational Linguistics and Intelligent Text Processing*, 3406:486–497.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT*, pages 347–354.

Yan, X., Han, J., and Afshar, R. (2003). CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of SDM*, pages 166–177.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of ICML*, pages 412–420.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of ACL*, pages 947–953.

Yu, B., Kaufmann, S., and Diermeier, D. (2007). Ideology classifiers for political speech. *Social Science Research Network Working Paper Series*.

Zelenko, D., Aone, C., and Richardella, A. (2003). Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Zhang, B., Hutchinson, B., Wu, W., and Ostendorf, M. (2010). Extracting phrase patterns with minimum redundancy for unsupervised speaker role classification. In *Proceedings of NAACL-HLT*, pages 717–720.

Zhang, B., Marin, A., Hutchinson, B., and Ostendorf, M. (2011). Analyzing conversations using rich phrase patterns. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 443–448.

Zhang, B., Marin, A., Hutchinson, B., and Ostendorf, M. (2013). Learning phrase patterns for text classification. *IEEE Transactions on Audio Speech and Language Processing*, (6):1180–1189.

Zhang, B. and Ostendorf, M. (2012). Semi-supervised learning for text classification using feature affinity regularization. In *Proceedings of Symposium on Machine Learning in Speech and Language Processing (MLSLP)*.

Zhang, C., Zuo, W., Peng, T., and He, F. (2008). Sentiment classification for chinese reviews using machine learning methods based on string kernel. pages 909–914.

Zhang, L., Zhang, D., Simoff, S. J., and Debenham, J. (2006). Weighted kernel model for text categorization. In *Proceedings of the fifth Australasian conference on Data mining and analystics - Volume 61*, pages 111–114.

Zhang, X. and Zhu, X. (2007). A new type of feature – loose n-gram feature in text categorization. In *Pattern Recognition and Image Analysis*, volume 4477, chapter 49, pages 378–385.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.