FEATURE SELECTION FOR SENTIMENT ANALYSIS
BASED ON CONTENT AND SYNTAX MODELS

A Thesis

Presented to

The Faculty of Graduate Studies

of

The University of Guelph

by

ADNAN DURIC

In partial fulfilment of requirements

for the degree of

Master of Science

May, 2011

© Adnan Duric, 2011

Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-82807-6
Our file Notre référence
ISBN: 978-0-494-82807-6

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

ABSTRACT

FEATURE SELECTION FOR SENTIMENT ANALYSIS BASED ON CONTENT AND SYNTAX MODELS

Adnan Duric
University of Guelph, 2011

Advisors:
Dr. Fei Song

Recent solutions for sentiment analysis have relied on feature selection methods ranging from lexicon-based approaches where the set of features are generated by humans, to approaches that use general statistical measures where features are selected solely on empirical evidence. The advantage of statistical approaches is that they are fully automatic, however, they often fail to separate features that carry sentiment from those that do not. In this thesis we propose a set of new feature selection schemes that use a Content and Syntax model to automatically learn a set of features in a review document by separating the entities that are being reviewed from the subjective expressions that describe those entities in terms of polarities. By focusing only on the subjective expressions and ignoring the entities, we can choose more salient features for document-level sentiment analysis. The results obtained from using these features in a maximum entropy classifier are competitive with the state of the art machine learning approaches.

Acknowledgments

First and foremost I offer my sincerest gratitude to my supervisor, Dr Fei Song, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to his encouragement and effort. One simply could not wish for a better or friendlier supervisor.

I would like to thank the members of my Advisory committee, Dr. Yang Xiang and Dr. Paul McNichollas for taking the time to review my written work and my presentations. Their knowledge in their respective fields helped me gain a more solid and well-rounded understanding for my own work.

I would also like thank my Examination committee, Dr. Stefan Kremer and Dr. Blair Nonnecke for evaluating my thesis, and for making sure that it meets a higher standard.

From an administrative standpoint, having a graduate secretary as friendly, supportive, and detail-oriented as Sheryl Beauchamp has helped me immensely in many aspects along the way.

In my daily work I have been blessed with a kind and knowledgeable group of fellow students. Alex “Xin” Li has provided me with something intangible throughout the years: A friendly smile and a “Hello” every morning. He has always encouraged me to keep going through my most difficult periods. Our friendship has been invaluable. Will Darling has been indispensable when reviewing my work and giving me helpful pointers along the way. He has also been the one with whom I could bounce ideas and talk with candidly about any trouble I was having. Most importantly, he has been a valuable friend.

Lastly, I would like to thank my wonderful family and friends who have supported me during the completion of my degree.

Table of Contents

List of Tables	iv
List of Figures	v
List of Algorithms	vi
1 Introduction	1
1.1 Applications	2
1.2 General Challenges	4
1.3 Existing Solutions	6
1.4 Motivations	7
1.5 Contributions	8
1.6 Overview of Document Structure	9
2 Background	10
2.1 Text Classification	10
2.1.1 Naive Bayes Classifiers	11
2.1.2 Maximum Entropy Classifiers	13
2.1.3 Discussion	19
2.2 Topic Modelling	20
2.2.1 Probabilistic Latent Semantic Indexing (pLSI)	20
2.2.2 Latent Dirichlet Allocation	22
2.2.3 Variational Inference	23
2.2.4 Gibbs Sampling	24
2.2.5 Evaluating Topic Models	26
2.2.6 Hidden Markov Model LDA (HMM-LDA)	27
2.2.7 HMM-LDA Inference	29
2.3 Feature Selection	30
2.4 Sentiment Analysis	32
2.4.1 Feature Selection For SA	32
2.4.2 Classification Methods	36
2.4.3 Topic Modelling Approaches to SA	40
2.4.4 Discussion	42
3 Proposed Feature Selection Methods for Sentiment Analysis	44
3.1 Feature Selection (FS) for Sentiment Analysis	44
3.1.1 Problems with Lexicon-Based Approaches	44
3.1.2 Characteristics of Salient Features	48
3.1.3 FS Based on HMM-LDA	52
3.1.4 HMM-LDA Training Procedure	54

3.2	FS Based on Syntactic Classes	58
3.3	Semantic-Syntactic Overlap Reduction Techniques	60
3.3.1	FS Based on Set Difference Between Syntactic and Semantic Classes	61
3.3.2	FS Based on Max Scores of Syntactic Features	63
3.4	Selecting Syntactic Features Based on POS Categories	64
3.5	Discussion	65
4	Experiments and Results	69
4.1	Data Set and Evaluation	69
4.1.1	Corpus Statistics	69
4.1.2	Evaluation Procedure	70
4.1.3	Evaluation Metrics	71
4.2	Baseline Results	72
4.2.1	Pang et al.'s FS Based on Document Frequency (DF)	72
4.2.2	Extended FS Based on Pang et al.'s DF	73
4.2.3	FS Based on Parts-of-Speech (POS)	74
4.2.4	Baseline Results	75
4.3	HMM-LDA Training	76
4.3.1	Qualitative Analysis - Example Annotated by Classes	77
4.4	Selecting Representative Words for Each Class	80
4.4.1	Analysis for the Syntactic Features	81
4.4.2	Correlation between Document Frequencies and Probabilities	82
4.5	Overlap Analysis	86
4.6	Semantic-Syntactic Overlap Reduction Methods	88
4.6.1	Set Difference	88
4.6.2	Max Score Selections	89
4.7	POS-assisted Semantic-Syntactic Overlap Reduction Methods	90
4.7.1	Overlaps with POS Tags	91
4.7.2	FS Based on a Subset of Syntactic Classes	92
4.8	Discussion	94
5	Conclusions and Future Directions	96
5.1	Conclusions	96
5.2	Future Directions	98
	Bibliography	101

List of Tables

4.1	General statistics for the movie review corpus. <i>Average word length</i> is the total number of characters divided by the total number of words in the corpus; <i>average sentence length</i> is the total number of words divided by the total number of sentences; <i>average document length</i> is the total number of words divided by the number of documents; and the <i>Lexical Diversity score</i> is the average number of times each vocabulary item appears in a document. The table shows the averages and standard deviations for the related statistics.	70
4.2	Baseline Results with a different number of iterations. Each column represents a feature selection method.	75
4.3	Top 10 highest-probability words extracted by the HMM-LDA for every class. The first column is the semantic class and the other columns are the syntactic classes.	78
4.4	Results for the HMM-LDA FS method at 10, 25 and eval iterations.	81
4.5	Pearson correlation between word probability and document frequency per class.	85
4.6	Word overlaps between classes at $\eta = 0.9$. The bottom value for each column is the number of words in that particular class. The first column indicates W_{sem} and the first row indicates the overlap between each syntactic class and W_{sem}	87
4.7	Results for the Syntactic-Semantic class set difference method. Each row represents the accuracy achieved at the particular ϵ value.	89
4.8	Results for the Max Score Based FS method.	90
4.9	POS tag percentage per class for POS groups (JJ*, RB*, VB*, NN*)	92
4.10	First row: Mean POS tags per class. Second row: Tags per class Std. Dev. Third row: Total number of words per POS group.	92
4.11	Classes included after low percentage POS' classes removed.	93
4.12	Results for the POS-aided method by POS categories in the included set POS'	93
4.13	Result for the POS-aided Max-Score method by POS categories in the included set POS'	93
4.14	Top 25 features given Max Score algorithm (3.3.2) with all syntactic classes. The last column shows the polarity odds per feature.	95

List of Figures

2.1	Supervised Classification Framework	11
2.2	The graphical model for pLSI using plate notation. The shaded circles represent the <i>observable</i> variables and the non-shaded are the <i>latent</i> variables. The variable d gets repeated M times and the variables z and w are repeated $M \times N$ times.	21
2.3	The plate diagram for LDA. The only observed variable is w	22
2.4	The procedure for learning LDA by Gibbs sampling [37]	25
2.5	(a) The graphical model of the HMM extensions to LDA. In addition to each word being endowed a topic, each word also has a dependency on syntactic classes that themselves are conditionally chained. (b) Generating phrases is done by following <i>transitions</i> between classes and choosing topics based on their probability mixtures for the semantic class.	29
3.1	Diagram of the parsing and HMM-LDA training procedure.	54
3.2	HMM-LDA Feature Selection. 1 Semantic class and $ C - 1 $ syntactic classes are represented.	58
4.1	HMM-LDA Log-likelihood by the number of Gibbs iterations. The 3 dotted lines represent the “burn-in” iterations for each chain. The 3 solid lines after 1000 iterations represent the sampled chains.	77
4.2	HMM-LDA Log-likelihood by the number of Gibbs iterations (closeup) . .	79
4.3	The top 25 words in W_{sem} and W_{14} sorted by the probability. The second column is the document frequency. The third column is the probability. . .	83
4.4	The bars represent the document frequency of each word. The line represents the word probability.	84
4.5	The bars represent the document frequency of each word. The line represents the word probability.	85

List of Algorithms

1	Generalized Iterative Scaling	18
2	Improved Iterative Scaling	18
3	HMM-LDA Gibbs Sampler	57
4	Syntactic-Semantic Set difference	62
5	Max Scores of Syntactic Features	64
6	Selecting a Subset of Syntactic Classes by POS Categories	65

Chapter 1

Introduction

In recent years, Web 2.0 has exploded with user-generated platforms such as blogs, discussion forums, and social networks. Opinionated consumers have at their disposal unprecedented reach and power by which to share their brand experiences and opinions. Major companies are now beginning to realize that their consumers wield more influences than ever before and as a result are beginning to look at reviews of their products and services more closely. In fact, this phenomenon is not only related to companies that make products and services, but also public figures and celebrities. These entities can then respond to consumer insights and general public sentiments by monitoring the places that generate them, and ultimately gain an advantage that has so far only come about with an extensive and usually expensive consumer research campaign.

According to two surveys of more than 2000 American adults [18]: 81% of Internet users have done online research on a product at least once and between 73% and 87% report that reviews had a significant influence on their purchases. However, the sheer volume of user-generated opinion content has become so enormous that most companies and public figures have to spend a lot of time engaged in it to find an overall sentiment. Indeed, even fellow consumers spend a lot of time sifting through many reviews in order to find that one review that discusses features and matters that are important to them.

One way for organizing such data is *text classification*, which involves mapping documents into *topical* categories based on the occurrences of particular features. Senti-

ment Analysis (SA) can be framed as a text classification task where the categories are *polarities* such as *positive* and *negative*. However, the similarities end here. Whereas general text classification is concerned with features that distinguish different topics, sentiment analysis deals with features about subjectivity, affect, emotion, and points-of-view that *describe* a particular topic in terms of a *positive* or *negative* sentiment.

1.1 Applications

“What other people think” has always been an important piece of information for most of us when we look to make a decision. Web sites such as *rottentomatoes.com* and *epinions.com* come to mind when we want to see what other people think about various movies and products, respectively. However, these consumer-facing sites are just the tip of the iceberg when it comes to new and established applications that use SA in a variety of ways.

1. *Online Customer Reviews*

As stated, *rottentomatoes.com* and *epinions.com* offer their users a forum in which to solicit feedback and reviews of various movies and products. Moreover, some large consumer web sites such as *Amazon* and *BestBuy* offer the same capabilities for their users for an enormous quantity of products. However, consumers are then faced with another problem. Reading reviews from other users is a very time-consuming task. Moreover, many reviews are conflicting. With this wealth of information, researchers are building automated systems that analyze this review data, learn from it and make predictions based on it. Effectively, the task of reading several reviews on a product or movie is reduced to simply observing an aggregate score.

2. *Sentiment Search*

Apart from augmenting the capabilities of existing systems, SA can also be integrated into a general-purpose search engine. Currently, when searching for products or movies in any commonly used search engine such as Google or Bing, the results consist of descriptions, company websites and related items. If the user is searching for reviews on a product or movie, she will undoubtedly have to navigate the results further. However, if the search engine knows that the user is searching for a review, it could bring up relevant review results closer to the top, and rank them according to positivity. It could even use SA techniques to automatically generate a summary or score based on the relevant information that it finds.

3. *Marketing and Business Intelligence*

Not only does SA have an ability to quickly aggregate, organize and summarize all kinds of user reviews and present the results to other consumers, it can also be used in the other direction to add value to the producers. In order for companies to remain competitive, they invariably must research the tastes of their customers very carefully. Instead of spending a lot of time sifting through customer reviews of their products, companies can use SA techniques to quickly aggregate and organize reviews into information that can help them learn about the relationship between their customers and their products. Another important area that companies have to consider when marketing their products is to target their products to consumers where the probability of conversion is the highest. Since there are many blogs that utilize contextual ad systems such as Google AdWords, the connection between a blog post about a product and the product advertisement should be as relevant as possible. For example, it does not make sense for a product to be advertised on a blog in which the review of that particular product is negative. By the same token,

if a blog is reviewing a product favourably, the ads should target that product as much as possible to increase the likely conversion rate.

4. *Detection of Inflammatory Text and Cyber-Bullying*

Perhaps an application area that is not immediately obvious for SA is to detect overly heated or antagonistic language. However, since at its heart, SA deals with opinions about products, places, things, it can readily deal with opinions about other people. Inappropriate remarks about individuals can be analyzed and eventually removed using SA techniques. As social media websites such as Facebook and Twitter are gaining more and more popularity, especially with children, cyber-bullying will only be a greater problem. Parents are rightly concerned that their children's on-line presence could be misused and slandered by their peers. SA techniques could provide valuable methods to help combat this recent phenomenon.

1.2 General Challenges

As we know, text classification seeks to classify a document by topic. As an example, Lewis et al. [22] attempt to categorize a large collection of Reuters news articles by topics such as *politics*, *sports*, *entertainment*, etc. In such approaches, documents are classified using word frequencies, so if a document has a high frequency of certain key words like 'football', there is an increased probability that the document is about *sports*.

However, since SA deals with opinions about topics and not topics themselves, the classification approaches with respects to both of the defined tasks in section 1.3 are necessarily different.

First of all, the indicators of sentiment must be aligned to the language that a reviewer is most likely to use when writing a review. Early work [14, 15] has focused

on using adjectives such as ‘*good*’ and ‘*bad*’ and adverbs like ‘*terrifically*’ and ‘*hatefully*’ as the important indicators of sentiment. Intuitively, this is what we would expect of an opinionated document. However, later research [19, 27, 30, 38] also suggests that other parts of speech such as verbs and even nouns [31] could be valuable indicators of sentiment.

The second difference between SA and topical text classification is how the indicators are combined to infer the polar category (either positive or negative) of a document. Pang et al. [30] shows that using the frequencies of the indicators (as it is done in topical text classification) has a negative impact. Therefore, they propose a scheme based on *presence* (whether an indicator occurs in the document or not). Moreover, it seems that subjective indicators occur less frequently than topical indicators [39], thus re-affirming the presence-based approach.

Thirdly, opinion and subjectivity are quite domain dependant. This is a little counter-intuitive because we tend to describe a wide array of things using the same words. For example, a movie can be described as ‘*great*’, and so can a product. However, there are examples where the same word offers different meanings in different domains. One such example is the word: ‘*unpredictable*’ [35]. It has positive connotations when referring to a movie, but negative ones when dealing with a product.

Finally, since user-generated documents are written using free-form natural language and tend to be rather informal with only liberal use of proper spelling and grammar, the task of somehow extracting the salient information, representing it and finally making predictions based on it is ultimately challenging.

1.3 Existing Solutions

The task of SA can be divided into Feature Selection and Document Classification. Feature Selection deals with specifically identifying the parts of a document that will most likely infer whether that document is *positive* or *negative*. Document Classification then combines these features of the document in ways that increase the odds of the document falling under one of these two polar categories.

Recent solutions for SA have relied on feature selection methods ranging from lexicon-based approaches where the set of features are generated by humans, to approaches that use general statistical measures where features are selected solely on empirical evidence.

Common approaches for feature selection, as used by researchers such as Das and Chen [9], start with a manually created lexicon specific to their particular domain whereas others [20,43] attempt to craft a general-purpose opinion lexicon that can be used across different domains. More recent lexicon-based approaches [10,19,21,31] begin with a small set of ‘seed’ words and bootstrap this set through synonym detection or various on-line resources to obtain a larger lexicon.

Many statistical feature selection methods for topical text classification can also be used for SA. The simplest statistical approach for feature selection by Pang et al. [30] is to use the most frequently-occurring words in the corpus as polarity indicators. This approach is commonly used with general text classification, and the results achieved indicate that simple document frequency cutoffs can be an effective feature selection scheme.

1.4 Motivations

The main advantage of statistical approaches for feature selection, such as the one by Pang et al. [30] is that they are fully automatic. However, they often fail to separate features that carry sentiment from those that do not. This is because words that are used frequently are not guaranteed to be subjective. In fact, it is usually the topical words that are repeated more often. Moreover, according to Pang and Lee [29], most reviews tend to use many different words for sentiment. Therefore, such approaches can pick up on topical words that do not carry any subjectivity.

On the other hand, humans generate lexicons based on the expressiveness of each feature so that each feature indeed carries sentiment. However, lexicon-based approaches have several key difficulties. First, they take time to compile. Whitelaw et al. [38] report that their feature selection process took 20 person-hours, since it involves work done by human annotators. In separate qualitative experiments done by [30], [40] and [21], the agreement between human judges when given a list of sentiment-bearing words is as low as 58% and no higher than 76%. In addition, some words may not be frequent enough for a classification algorithm.

Clearly, a feature selection method that utilizes the advantages of both approaches while minimizing their disadvantages is desired. Specifically, we can think of a review document as a mixture of topical words, referred to as *entities*, that do not carry any sentiment, and modifying words that relate a certain entity with a polarity. Therefore, the task of classifying a review document can be explored by taking into account this mixture and separating out the entities in favour of the modifiers.

In this thesis we propose a set of new feature selection schemes that use a Content and Syntax model [13] to automatically learn a set of features in a review document

by separating the entities that are being reviewed from the subjective expressions that describe those entities in terms of polarities. By focusing only on the subjective expressions and ignoring the entities, we can choose more salient features for document-level sentiment analysis.

1.5 Contributions

The major contributions in this thesis are:

1. *Characteristics of Salient Features*

Based on our intuition in regards to the types of features that are most useful for SA, and backed up by experimental evidence from our approach as well as established approaches, we describe a list of the most important characteristics that need to be considered when selecting features for SA.

2. *Feature Selection*

We describe and analyze a set of new feature selection schemes that *automatically* select some of the most important sentiment indicators from a collection of opinionated documents. Our approaches are based on a Content and Syntax model that can separate content words about entities from syntactic words that modify the entities.

3. *Comparison of Established and Proposed Methods*

Finally, we carry out qualitative and quantitative analyses of established approaches in relation to our solutions by examining the classification accuracies obtained for each.

1.6 Overview of Document Structure

The rest of this thesis is organized as follows. Chapter 2 presents a review of the background literature in the fields of text classification, feature selection, sentiment analysis, topic modelling, and machine learning. It provides the relevant background information needed for our own approaches as well as explaining the related solutions and state-of-the-art methods encountered for each subject. Chapter 3 describes our proposed solutions and details our implementation. We first outline a set of characteristics that we follow to select salient features, and based on those characteristics, propose a set of feature selection schemes as well as a way of using the selected features to classify review documents. Chapter 4 explains our experimental design, results, and analyses. We use the assessments of three baseline approaches to quantify our own approaches in terms of document classification accuracy. Finally, Chapter 5 concludes and points out some future directions for this line of research.

Chapter 2

Background

2.1 Text Classification

Text classification is concerned with automatically assigning a document into a specific category based on observable patterns of features. It generally involves two common tasks:

1. The identification of features that are salient for distinguishing documents.
2. The construction of models that perform text classification automatically.

The first is referred to as Feature Selection, and some important approaches will be reviewed in section 2.3. The second is about choosing the correct category for a given document.

In the basic classification task, each input is considered in isolation from all other inputs, and the set of labels is defined in advance. Some example classification tasks are:

- Deciding whether an email is a spam or not (spam detection).
- Deciding what the topic of a news article is from a fixed list of topics such as “sports,” “technology,” and “politics” (topical text classification).
- Deciding whether a given occurrence of the word ‘bank’ refers to a river bank, a financial institution, the act of tilting to the side, or the act of depositing something in a financial institution (word sense disambiguation).

A classifier is said to be *supervised* if it is built from training data with the correct labels for each input. The framework for supervised classification is shown in Figure 2.1. The process is split up into *Training* and *Prediction* phases. Before the training phase, the input data are converted into features and associated with an output category, forming a feature set. During training, the feature sets and categories are input into a machine learning algorithm to generate a classifier. During prediction, the classifier outputs a category for each new input based on the model learned in the training phase.

Figure 2.1: Supervised Classification Framework

In this thesis, we review two different machine learning algorithms that are prominently used for supervised text classification: Naive Bayes classifiers and Maximum Entropy classifiers.

2.1.1 Naive Bayes Classifiers

In Naive Bayes classification [1], the contribution from each feature is used to calculate the probability of assigning a category label to an input. This contribution is then combined with the *prior probability* of each label to determine its likelihood estimate.

The prior probability is defined as the empirical frequency of each label in the training set. Then, the label with the highest likelihood estimate is assigned to the given input.

The naive Bayes classifier chooses the most likely label for an input, under the assumption that every input value is generated by first choosing a class label for that input value, and then generating each feature, entirely independent of every other feature. However, treating each feature independently is often overly simplistic since features can indeed be dependent on one another. In practise, exploiting this *independence assumption*, known as the *naive Bayes* assumption works well for many complex real-world problems by making it easier to combine the contributions of the different features in a computationally efficient way.

Using this assumption, the probability model for the classifier is represented as a conditional distribution, $P(\text{label}|\text{features})$, stating the probability that an input will have a particular category label given a feature set. The maximum likelihood estimate is obtained by choosing a label for a new input that maximizes $P(l|\text{features})$, for each l category label. From the definition of conditional probability, we have:

$$P(\text{label}|\text{features}) = P(\text{features}, \text{label})/P(\text{features}) \quad (2.1)$$

The denominator, $P(\text{features})$ is the same for all labels since the values of the features is given, so it simply suffices to compute the *label likelihood* $P(\text{features}, \text{label})$ to find the most likely label.

Using Bayes' Theorem, the label likelihood can be expanded into the probability of the label times the probability of the features given the label:

$$P(\text{features}, \text{label}) = P(\text{label}) \times P(\text{features}|\text{label}) \quad (2.2)$$

Using the independence assumption, the joint probability can be rewritten as follows using the *chain rule*

$$P(features, label) = P(label) \times \prod_{f \in features} P(f|label) \quad (2.3)$$

where $P(f|label)$ is the contribution of a feature f toward the likelihood for a label. This can be approximated from the relative frequencies in the training set

$$P(f|label) = N(f, label)/N(label) \quad (2.4)$$

where $N(f, label)$ represents the counts of a feature and label co-occurring and $N(label)$ is the total number of labels

This is the *Maximum Likelihood Estimate*, since all the probability mass is given to the training set. However, if a feature and category label pair never occur together in a training set, then the relative frequency for the pair will be *zero*. This is problematic because the resulting multiplication of the entire feature space will also result in zero. To alleviate this problem, there are various *smoothing* techniques, the simplest of which adds 1 to each $N(f, label)$, and is called *Laplace smoothing*.

For a more detailed discussion on Naive Bayes classifiers, the reader is referred to [24]

2.1.2 Maximum Entropy Classifiers

The Maximum Entropy classifier [1] uses a model that is similar to that for a naive Bayes classifier. The main difference is that instead of using probabilities to represent the parameters, it iteratively searches for a set of parameters to maximize the likelihood of a label given the feature set, and the total likelihood of the training corpus

This is defined as

$$P(features) = \sum_{x \in corpus} P(label(x)|features(x)) \quad (2.5)$$

where $P(label|features)$ is the probability that an input whose features are *features* will have class label *label*, is defined as

$$P(label|features) = \frac{P(label, features)}{\sum_l P(l, features)} \quad (2.6)$$

Since the independence assumption no longer applies, the feature space could potentially have many complex interactions, and obtaining a maximum likelihood estimate of the training set is *intractable*. As a result, Maximum Entropy classifiers model the feature space parameters using *iterative optimization* techniques. This optimization process of searching for a parameter's optimum value proceeds by randomly initializing it and repeatedly refining it, until an optimal solution has been found. At each step, the parameters are guaranteed to get closer to their optimal values, however, there are no means of determining whether the optimal values have been reached. Instead, we must rely on a set of *convergence conditions* that involve the number of iterations and / or changes in estimated parameter values. As such, these iterative optimization techniques generally take a long time to converge, where the time is proportional to the size of the training set, the size of the feature space, and the number of category labels.

Maximum Entropy Model

The Maximum Entropy classifier is similar to the naive Bayes classifier in terms of how the likelihood for each category label is calculated in the feature space. It defines the prior probability for each parameter based on empirical evidence for each feature and

multiplies each feature-category pair contribution towards a category label's likelihood. Whereas the naive Bayes classifier assigns the same value to the contributions of all the features, the Maximum Entropy classifier can decide the combinations of labels and features that should receive their own values. The outcome of this is that it is possible to use a single parameter to associate a given feature with more than one label, or alternately, to associate a given label with more than one feature.

Therefore, each one of these selected combinations of labels and features that are associated with a parameter in the Maximum Entropy model are unique *joint-features*. Each category label defines a joint feature $w[label]$, and each feature-category pair defines a joint feature $w[f, label]$. In this case, the difference between a joint-feature and a (simple) feature is that the joint-feature is a feature-category pair, derived from the training set, while a feature is not associated with a category label. Thus, for each input, the contribution from each label is given by the product of the parameters of the joint-features that apply to that input and category label:

$$P(input, label) = \prod_{joint-features(input, label)} w[joint - feature] \quad (2.7)$$

Feature Functions

There are two main types of feature functions. For *binary* feature functions, the weights are calculated by the presence or absence of a joint-feature [24]. More formally, let $f_{i,c'}(x, c)$ be a joint-feature for word i and category label c' :

$$f_{i,c'}(x, c) = \begin{cases} 1 & \text{if } c = c' \text{ and } x \text{ contains word } i \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

The Maximum Entropy approach is not limited to binary features and the word counts for a particular document can also be used as a weight [24]. Such features would be defined as:

$$f_{i,c'}(x, c) = \begin{cases} 0 & \text{if } c \neq c' \\ \frac{N(x, i)}{N(x)} & \text{otherwise} \end{cases} \quad (2.9)$$

Loglinear Model

Employing the feature functions described, the Maximum Entropy Loglinear model takes the following form:

$$P(x, c) = \frac{1}{Z} \prod_{i=1}^K \alpha_i^{f_i(x, c)} \quad (2.10)$$

where K is the number of features, α_i is the weight of feature f_i and Z is a normalizing constant, which is calculated by summing the probability contributions from all classes:

$$Z = \sum_{c=1}^C \prod_{i=1}^K \alpha_i^{f_i(x, c)} \quad (2.11)$$

where $c \in C$ is a category. Taking the *log* on both sides, we derive the following loglinear form:

$$\log P(x, c) = -\log Z + \sum_{i=1}^K f_i(x, c) \log \alpha_i \quad (2.12)$$

Parameter Estimation

Maximum Entropy Models must obey the following constraints based on the training data such that each constraint expresses a characteristic of the training data that should also

be present in the learned distribution:

$$Ep^*f_i = E\tilde{p}f_i \quad (2.13)$$

for a Maximum Entropy model distribution p^* and the *empirical* distribution $\tilde{p}$. The empirical distribution is calculated from the training data as follows:

$$E\tilde{p}f_i = \frac{1}{N} \sum_{j=1}^N f_i(x_j, c(x_j)) \quad (2.14)$$

where N is the number of elements in the training set.

The Expectation of the computed probability distribution is defined as:

$$Ep f_i = \sum_{x,c} p(x, c) f_i(x, c) \quad (2.15)$$

which is generally *intractable* since it would involve summations over all possible values of x and c . However, using only empirically observed x , this distribution can be approximated:

$$Ep f_i = \sum_{x,c} \tilde{p}(x) p(c|x) f_i(x, c) = \frac{1}{N} \sum_{j=1}^N \sum_c p(c|x_j) f_i(x_j, c) \quad (2.16)$$

where $c \in C$ is a category.

There are two *iterative optimization* algorithms that provide initial settings for model parameters and repeatedly refine these parameters to bring them closer to an optimal solution.

The first algorithm is called *Generalized Iterative Scaling*, which requires that the sum of the features for each possible combination (x, c) be equal to a constant C . In order to fulfil this requirement, we define C as the greatest possible feature sum over *all*

possible data.

$$C = \max_{(x,c)} \sum_{i=1}^K f_i(x, c) \quad (2.17)$$

and add a filler feature f_{K+1} that is defined as follows.

$$f_{K+1}(x, c) = C - \sum_{i=1}^K f_i(x, c) \quad (2.18)$$

Algorithm 1 Generalized Iterative Scaling

```

1  $n \leftarrow 1$ 
2 Initialize  $\alpha_i$  with 1 Compute  $E\tilde{p}f_i$  (from 2.14).
3 repeat
4   Compute 2.16 with current  $\alpha_i$  for all  $i$ 
5   Update parameters  $\alpha_i$  by.  $\alpha_i^{(n+1)} = \alpha_i^{(n)} (\frac{E\tilde{p}f_i}{Ep^*f_i})^{\frac{1}{\tilde{c}}}$ 
6    $n \leftarrow n + 1$ 
7 until all parameters have converged
```

The *Improved Iterative Scaling* algorithm is a variation of Generalized Iterative Scaling that does not require the filler feature function.

Algorithm 2 Improved Iterative Scaling

```

1 Initialize  $\lambda_i$  to 0 where  $1 \leq i \leq K$ . Compute  $E\tilde{p}f_i$  (from 2.14)
2 repeat
3   Compute 2.16 with current  $\lambda_i$  for all  $i$ 
4   Compute:  $\delta_i = \log \frac{E\tilde{p}f_i}{Ep^*f_i}$ 
5   Update the parameters  $\lambda_i$   $\lambda_i^{(n+1)} = \lambda_i^{(n)} + \delta_i$ 
6    $n \leftarrow n + 1$ 
7 until all parameters have converged
```

Convergence can be attained by setting a maximum number of iterations prior to training, or when the sum of the differences in the values of the parameters between iterations is below a certain threshold.

2.1.3 Discussion

In our approach to Sentiment Analysis, we use Maximum Entropy (ME) for text classification. The motivating idea behind ME is that the probability distribution that best describes a model subject to some constraints is the one that takes into account those constraints while assuming nothing else about the model, effectively maximizing the *entropy* of the parts of the model that are not under any constraints.

In the previous chapter, we introduced some challenges for SA. There are often a large number of low frequency terms that can be important, and the training data tends to be rather small. ME deals with these issues simultaneously by basing its classification decisions on only the data that it *empirically* observes, and at the same time avoids *scaling* down the other data in the direction of more frequent, but less important features [24].

The main advantage that ME has over the Naive Bayes (NB) model is that it does not assume feature independence. For words such as “Buenos Aires” that almost exclusively appear together, NB would treat them as independent when they are clearly not [26].

The biggest disadvantage of ME modelling is that it is susceptible to *overfitting*. This means that the parameters can be trained to behave like the training data to such an extent that the classification of unseen data can suffer. Limiting the number of training iterations is important, but since it is almost impossible to guess the right number of iterations given a data and feature set, we often use a *validation* set to control the search process (see subsection 4.2.2).

2.2 Topic Modelling

Topic models are statistical models that aim to discover *latent* topics in a document or a document collection. They are based upon the idea that a document is a *mixture* of topics, each of which is a probability distribution over all the words in the *vocabulary* (the set of all unique words) [5, 12, 17, 33]. The resulting topics are groups of words that have similar *semantic* meaning. Note that this definition of ‘topic’ is different from the definition used for text classification, which signifies the relevant output category for a document.

There is a fundamental *assumption* in regards to topic models [5]. Since these models use the “bag-of-words” approach (where the word frequency is considered but the order is irrelevant), where the word order in the document is irrelevant, in probability theory we can say that these words are *exchangeable*. We can further assume that documents in a corpus are exchangeable. According to De Finetti’s Theorem [5], any sequence of exchangeable random variables can be represented by a mixture model. Thus, we seek mixture models to capture this exchangeability for both words and documents.

2.2.1 Probabilistic Latent Semantic Indexing (pLSI)

The first topic model, developed by Hoffman [17] and known as *Probabilistic Latent Semantic Indexing (pLSI)*, uses ideas from *Latent Semantic Indexing (LSI)*. LSI is a technique in natural language processing that analyzes the relationships between a set of documents and the terms in those documents to provide a set of *latent groups* whereby similar words are clustered together

The pLSI approach models each word in a document as a sample from a *mixture model*. The components of this mixture model are a set of *multinomial* random variables

that can be thought of as ‘topics.’ Each document specifies the mixing proportions for these components. Thus, each word is generated from a single topic, and different words are generated from different topics.

More formally, if we let z_k be one of K topics, a document d and word w_n are *conditionally independent* given an *unobserved (latent)* topic z :

$$p(d, w_n) = p(d) \sum_{k=1}^K p(w_n | z_k) p(z_k | d) \quad (2.19)$$

We can visualize the conditional dependencies of the variables in this model using the *Plate notation* in Figure 2.2. It is a method of representing variables that are repeated in a graphical model. A plate (rectangular shape) is used to group variables that are repeated together. The number of repetitions is usually specified in the bottom-left corner of a plate.

Figure 2.2: The graphical model for pLSI using plate notation. The shaded circles represent the *observable* variables and the non-shaded are the *latent* variables. The variable d gets repeated M times and the variables z and w are repeated $M \times N$ times.

pLSI is an incomplete model with several difficulties. Each document is represented by mixing proportions of topics, but there is no probabilistic model for these proportions. Therefore, the number of parameters grows *linearly* with the size of the corpus and there is no direct way to assign document probabilities outside of the training

set. These two difficulties lead to overfitting problems and make inferences impossible [5].

2.2.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a *generative probabilistic model* that is similar to pLSI in that documents are mixtures over latent topics that are themselves distributions over words. However, it is a full probabilistic generative model and we can reverse the generative procedure to do inferences on unseen data.

Figure 2.3: The plate diagram for LDA. The only observed variable is w .

The LDA generative process can be stated as follows:

For each document d in corpus D :

1. Sample θ from a Dirichlet prior $Dir(\alpha)$
2. For each word w_n in document d :
 - (a) Draw a topic $z_n \sim Multinomial(\theta)$
 - (b) Choose a word $w_n \sim Multinomial(\beta_{z_n})$

where β is a Dirichlet [11] prior for a $V \times K$ dimensional matrix ϕ such that $\sum_{v=1}^V \phi_{k,v} = 1$ for $1 \leq k \leq K$, and α is a K -dimensional vector such that $\alpha = (\alpha_1, \alpha_2, \alpha_3 \dots \alpha_K)$ and

$\alpha_k > 0$ for $1 \leq k \leq K$.

As shown in Figure 2.3, the three levels of LDA are the corpus level parameters α and β which are sampled once for the corpus, the document level variables θ which are sampled for each document, and the word level variables z and w which are sampled for each word in the document.

The full joint distribution for a document given the priors α and β is:

$$p(\theta, \phi, z, w | \alpha, \beta) = p(\theta | \alpha) \prod_{k=1}^K p(\phi_k | \beta) \prod_{n=1}^N p(z_n | \theta) p(w_n | \phi_{z_n}) \quad (2.20)$$

Comparing with the pLSI model, it is clear that LDA overcomes the problems of pLSI because it is sampling the topic mixture from a K -parameter Dirichlet prior distribution rather than having parameters linked directly to the training set. This means that LDA does not depend on the size of the training set and does not suffer from the overtraining issues. Also, since it is a full generative model, we can establish an inference procedure for unseen data.

2.2.3 Variational Inference

Since LDA is a Bayesian model, we can use Bayesian inference to determine the posterior distribution of the latent variables. Bayesian inference is a method of statistical inference where a *posterior* is calculated from a *prior* and a *likelihood*: $\text{posterior} \propto \text{prior} \times \text{likelihood}$. Therefore, the key inferential problem for LDA is given by:

$$p(\theta, \phi, z) = \frac{p(\theta, \phi, z, w | \alpha, \beta)}{p(w | \alpha, \beta)} \quad (2.21)$$

where the denominator is calculated by marginalizing out the θ , ϕ and z :

$$p(w|\alpha, \beta) = \int_{\theta} \int_{\phi} \sum_z p(\theta, \phi, z, w|\alpha, \beta) d\phi d\theta \quad (2.22)$$

This function is *intractable* and cannot be computed directly, but there are two *approximate inference* methods that can be used. Both *variational* inference and *Gibbs sampling* approaches exploit the *conjugate* relationship between the Multinomial and Dirichlet distributions. Briefly, two distributions are conjugate if when multiplied together, the same form is exhibited as one of the distributions. In this case, when a Multinomial is multiplied with a Dirichlet, the result is another Dirichlet.

Since we will not be using variational inference from this point in the thesis, we refer the reader to [5] for a full derivation of an iterative approach based on the *Expectation Maximization (EM)* algorithm [2].

2.2.4 Gibbs Sampling

An alternative method of inference for LDA is via *Gibbs sampling*. This differs from the variational inference approach because we are not explicitly estimating θ and ϕ but rather derive a sampling algorithm that samples from the *posterior* distribution over the assignment of words to topics, $p(z|w)$ [12, 33].

Gibbs sampling is a form of *Markov Chain Monte Carlo (MCMC)* methods, which refers to a class of approximate techniques that iteratively ‘sample’ values from complex distributions using simpler *conditional* distributions. Another way of describing this technique is to think about the sampling process itself, which sequentially samples a value from each variable conditioned on all the other variables and proceeds until the sampled values approximate a *target* distribution [33].

This means that in order to sample from $p(z|w)$, we need the *full conditional posterior*

Figure 2.4: The procedure for learning LDA by Gibbs sampling [37]

distribution $p(z_i | z_{-i})$ where z_{-i} denotes all topics z_j where $j \neq i$.

From the joint distribution defined in 2.20, we can see that $p(w|\phi)$ and $p(\phi|\beta)$ depend only on ϕ , and $p(z|\theta)$ and $p(\theta|\alpha)$ depend only on θ . This means that we can *collapse* the Gibbs sampler by *integrating* out ϕ and θ separately giving:

$$p(w|z, \beta) = \int p(w|z, \phi) p(z|\alpha) d\phi \quad (2.23)$$

and

$$p(z|\theta) = \int p(z|\theta) p(\theta|\alpha) d\theta \quad (2.24)$$

Using the functional forms of the Dirichlet and Multinomial distributions and substituting them into 2.20, 2.23, 2.24, we get [33]:

$$p(z_i = j | z_{-i}, w_i, d_i) \propto \frac{C_{w_i j}^{WT} + \beta}{\sum_{w=1}^W C_{w j}^{WT} + W\beta} \times \frac{C_{d_i j}^{DT} + \alpha}{\sum_{t=1}^T C_{d_i t}^{DT} + W\alpha} \quad (2.25)$$

where C^{WT} is a matrix of word frequencies per topic of size $W \times T$ and C^{DT} is a matrix of topic frequencies per document of size $D \times T$. For a fuller and more detailed treatment of the derivation of the Gibbs sampler, the reader is referred to [37].

The sampler described in 2.25 gives direct estimates of z , which are the topic

assignments for each word. From there we can derive the estimates of ϕ and θ , the word-topic distribution and topic-document distribution, respectively.

$$\phi_i^j = \frac{C_{w_i,j}^{WT} + \beta}{\sum_{w=1}^W C_{w_j}^{WT} + W\beta} \quad (2.26)$$

$$\theta_j^d = \frac{C_{d_i,j}^{DT} + \alpha}{\sum_{t=1}^T C_{d_i,t}^{DT} + W\alpha} \quad (2.27)$$

These values are the *posterior means* conditioned on a particular z . This implies that for *unseen* documents, ϕ and θ are used as the *predictive* distribution of sampling a new word i from topic j in document d .

2.2.5 Evaluating Topic Models

The classical method of evaluating the generalizability of a topic model is *perplexity*. Perplexity indicates *uncertainty* in the *unseen* data: the lower a perplexity value the better. If a topic model is able to capture the trends in a data set well, then it should be quite capable of predicting withheld data. As indicated by Heinrich [16], the perplexity for a topic model is defined as:

$$p(\tilde{w}|M) = \prod_{m=1}^M p(\tilde{w}_m|M)^{-\frac{1}{N}} = \exp\left(-\frac{\sum_{m=1}^M \log p(\tilde{w}_m|M)}{\sum_{m=1}^M N_m}\right) \quad (2.28)$$

where $\tilde{w}$ is the corpus of unseen test data, M is the *state* of the model, and N is the number of words in test document $\tilde{w}_m$. In the case of LDA, M consists of the topic-document and topic-word distributions θ and ϕ , respectively.

However, evaluating topics is not as simple as calculating the perplexity of a topic model. Indeed, since topic discovery relies on word co-occurrences, there can be no

simple quantitative method to assign a ‘goodness’ value to a particular topic given the words that it contains. What might better explain a corpus to a model may not match the explanations, concepts, or level of semantics that a human would infer.

This can be seen in the research done by Chang et al. [7] where the authors carried out a series of qualitative experiments with human participants on the assumed coherence and relevance of topic models. They came to the conclusion that traditional metrics, such as perplexity, are *negatively* correlated with the measurements of topic quality given by the participants. In other words, having a low perplexity does not translate into having better topics based on human judgement. Therefore, the best measure of topic quality is subject to the task at hand.

2.2.6 Hidden Markov Model LDA (HMM-LDA)

The Hidden Markov Model LDA (HMM-LDA) [13] is a generative topic model built on top of LDA that simultaneously models topics and syntactic classes in a collection of documents. The idea behind the model is that a typical word can play different roles. A word can either be part of the *content* and serve in a *semantic* (topical) purpose or it can be part of the grammatical (*syntactic*) scheme. It can also be used in both contexts.

HMM-LDA models this behaviour by inducing *syntactic classes* for each word based on how they appear together in a sentence using a *Hidden Markov Model*. Each word gets assigned to a syntactic class, but one class is reserved for the semantic words. Words in this class behave as they would in a regular LDA topic model, having a certain probability of appearing in a document.

Some of the syntax words that appear in a document include adjectives, prepositions, pronouns, adverbs, verbs, and punctuations, while the majority of nouns can be thought of as semantic words. An important point to realize is that HMM-LDA models

different kinds of *dependencies* between words, therefore the syntactic classes do not necessarily map to particular parts-of-speech categories. These dependencies act as constraints whereby syntactic words are dependent on each other only in a short range, usually given by a sentence or a phrase, whereas semantic word dependency can cover a long range, or even a whole document. This is an important idea because it enables us to extract the important nouns that are associated with a particular topic, regardless of how they appear syntactically in individual sentences. Conversely, we can obtain the syntactic words that *modify* such nouns without needing to explicitly know about them. This dual capacity comes from the HMM component that models the syntactic relationships between the words in sentences and the LDA component that models the topical word mixtures. More formally, the model is defined in terms of three sets of variables and a generative process:

$\mathbf{w} = \{w_1, \dots, w_n\}$ is a sequence of words where each word w_i is one of V words.

$\mathbf{z} = \{z_1, \dots, z_n\}$ is a sequence of topic assignments where each z_i is one of K topics.

$\mathbf{c} = \{c_1, \dots, c_n\}$ is a sequence of class assignments where each c_i is one of C classes with $c_i = 1$ designated as the ‘semantic class’, and the rest, the ‘syntactic’ classes.

We have already seen the $\mathbf{w}$ and $\mathbf{z}$ vectors are used in the LDA model. The HMM-LDA model augments a third vector $\mathbf{c}$ that represents the syntactic class assignments for each word.

Since we are dealing with a Hidden Markov Model, we require a variable representing the *transition probabilities* between the classes. These probabilities are given by a $C \times C$ *transition matrix* π that models the transitions between classes c_{i-1} and c_i . The generative process is given as follows:

1. Sample $\theta^{(d)}$ from a Dirichlet prior $Dir(\alpha)$
2. For each word w_i in document d :

Figure 2.5: (a) The graphical model of the HMM extensions to LDA. In addition to each word being endowed a topic, each word also has a dependency on syntactic classes that themselves are conditionally chained. (b) Generating phrases is done by following *transitions* between classes and choosing topics based on their probability mixtures for the semantic class.

(a) Draw $z_i \sim \theta^{(d)}$

(b) Draw $c_i \sim \pi^{(c_i-1)}$

(c) If $c_i = 1$, then draw $w_i \sim \phi^{(z_i)}$, else draw $w_i \sim \phi^{(c_i)}$

where $\phi^{(z_i)} \sim \text{Dir}(\beta)$ and $\phi^{(c_i)} \sim \text{Dir}(\delta)$, both from *Dirichlet* distributions.

2.2.7 HMM-LDA Inference

As with the basic LDA model, the exact inference for HMM-LDA is intractable. However, inferences of this model can be done via Gibbs sampling where we iteratively draw a topic assignment z_i and a class assignment c_i for each word w_i in the corpus. Therefore, given the words $w_i \in \mathbf{w}$, the class assignments $c_i \in \mathbf{c}$, the topic assignments $z_i \in \mathbf{z}$, and all other topic assignments z_{-i} , we get a distribution of topics:

$$P(z_i | z_{-i}, c, w) \propto P(z_i | z_{-i}) P(w_i | z, c, w_{-i}) \quad (2.29)$$

$$P(z_i | z_{-i}, c, w) \propto \begin{cases} n_{z_i}^{(d_i)} + \alpha & \text{if } c_i \neq 1 \\ (n_{z_i}^{(d_i)} + \alpha) \times \frac{n_{w_i}^{(z_i)} + \beta}{n^{(d_i)} + W\beta} & \text{if } c_i = 1 \end{cases} \quad (2.30)$$

Similarly, we get a distribution of classes c_i conditioned on all the other classes c_{-i} :

$$P(c_i|c_{-i}, z, w) \propto P(w_i|z, c, w_{-i}) P(c_i|c_{-i}) \quad (2.31)$$

$$P(c_i|c_{-i}, z, w) \propto \begin{cases} \frac{n_{w_i}^{(c_i)} + \delta}{n^{(c_i)} + W\delta} \times \frac{(n_{c_i}^{(c_{i-1})})(n_{c_i+1}^{(c_i)} + I(c_{i-1}=c_i) \times I(c_i=c_{i+1}) + \gamma)}{n^{(c_i)} + I(c_{i-1}=c_i) + C\gamma} & \text{if } c_i \neq 1 \\ \frac{n_{w_i}^{(z_i)} + \beta}{n^{(z_i)} + W\beta} \times \frac{(n_{c_i}^{(c_{i-1})})(n_{c_i+1}^{(c_i)} + I(c_{i-1}=c_i) \times I(c_i=c_{i+1}) + \gamma)}{n^{(c_i)} + I(c_{i-1}=c_i) + C\gamma} & \text{if } c_i = 1 \end{cases} \quad (2.32)$$

where $n_{w_i}^{c_i}$ is the number of times word w_i is assigned to class c_i ; $n_{w_i}^{z_i}$ is the number of times w_i is assigned to topic z_i ; and n_{c_i} is the number of transitions from class c_{i-1} to class c_i .

2.3 Feature Selection

Feature Selection (FS) is a technique to select a subset of features which contribute the most to distinguishing between different classification categories. This is important in many text classification tasks because the potential feature set can be large. For example, in Pang et al.'s movie review corpus, there are approximately 40,000 distinct terms. If we were to include all the terms in our feature set, the classification task becomes computationally expensive, and the feature set is likely to contain many irrelevant features, or even features which contribute negatively to classification by making it difficult to separate certain categories.

A general FS procedure first ranks every potential feature according to some metric, and then removes all the features except some pre-determined number of top ranked features. In our FS schemes we use the *document frequency (DF)* feature selection

method to cut the number of features into a manageable number.

DF measures the general importance of a word in a feature set by counting the number of documents that the word appears in. More formally, the DF score for word w_i is calculated as follows:

$$df(w) = |\{d | w_i \in d\}| \quad (2.33)$$

where $d \in D$ is a document in the corpus.

There are two main advantages of the DF feature selection method over simply taking the *term frequency* of every features (how many times the feature appears in the whole corpus). First of all, the influence of the common terms such as stop words will be minimized because words cannot have a DF value that is higher than the number of documents. Moreover, it minimizes the impact of some words that could have a higher frequency in a subset of documents, but are generally not used throughout the corpus.

Other feature selection schemes utilize the *output* categories from the training set to select the most appropriate features. One such method is to use the *chi-square* (χ^2) [42] statistic to measure the independence between a word and an output category. This is done by using a 2-by-2 contingency matrix such that each word has a count of the combination of its presence in a document and a category. As such, features are found to be strongly dependent on the categories. Moreover, according to Cai [6], the χ^2 FS scheme ranks some rare features higher than more common ones.

Another FS scheme that attempts to capture the relationship between features and categories is the *Optimal Orthogonal Centroid* (OCFS) scheme [41]. The premise of this scheme is that each category has a centroid vector whereby the features are the dimensions. The process consists of iteratively removing features from the centroid vector

such that the distance between all category centroid vectors is maximized. The remaining set of features will be more discriminative in that they will be better able to distinguish between classes.

2.4 Sentiment Analysis

As described in the Chapter 1, the task of Sentiment Analysis (SA) can be divided into two stages, akin to general text classification.

1. Identifying the parts of a document (e.g., words, sentences, expressions) that are most likely related to its relevant category.
2. Combining these parts of the document in ways that increase the odds of mapping the document into its relevant category.

2.4.1 Feature Selection For SA

The simplest method for feature selection borrows the idea from general text classification of using a set of the most frequently occurring words as features. Pang et al. [30] compare unigrams (single words) and bigrams (two co-occurring words) that have been selected by the DF cutoff method (see section 2.3). They experiment with different DF *cutoff* ranges (how many unigrams / bigrams are selected into the final feature set). Overall, their most accurate result is with unigrams at a relatively conservative cutoff of 2633 most frequent unigrams.

Some methods can be thought of as closely following the common intuitions about salient sentiment features. They typically use a manually-compiled list of features that tend to reflect empirical evidence from experience of reading review documents. Such

lists are known as *lexicons* and they usually contain many modifiers such as adjectives and adverbs.

The lexicon approach can be seen in the “Sentiment Analyzer” [43] system. The researchers have collected sentiment words from several sources such as the “General Inquirer”, the “Dictionary of Affect of Language” and WordNet. In total, they have compiled a list of about 2500 adjectives and less than 50 nouns, affirming that adjectives are an important sentiment indicator. Their system works by extracting common patterns of *noun-phrases* (that most likely represent an object being reviewed) and use their lexicon to assign a polarity to these objects if the object and a word in the lexicon co-occur at the phrase or sentence level.

Other researchers found that targeting the lexicon to a specific domain results in higher accuracy at the expense of domain-dependence. Das and Chen [9] analyze the sentiment of financial data using a lexicon of language relating to financial reviews. They experiment with the idea of predicting stock *value* based on reviews about that stock with five different automatic classification schemes that utilize their lexicon. They recommend that if their approach is used for another domain, a new lexicon should be developed.

Something similar can be seen in Hurst and Nigam [20]. They harvest a collection of review messages from various online sources and classify these messages into different domains. They then hand label each sentence in the message from a domain by topic and polarity, using only the words that appear in the message. Thus each domain ends up having its own custom lexicon of sentiment-carrying words.

An interesting approach by Ding et al. [10] attempts to deal with the domain dependency problem of lexicons by proposing a series of linguistic conventions which use contextual information from other reviews to determine the context of the word in question. They give the example of “the battery life is very long.” Even though it is not

clear whether the word “long” is positive or negative, another review might have “the battery life is long, which is great!”

More recent lexicon-based approaches [10, 19, 21, 31] begin with a small set of ‘seed’ words and bootstrap this set through synonym detection or various online resources to obtain a larger lexicon. Riloff et al. [31] aim to discover subjective nouns by bootstrapping extraction patterns using a small set of *seed* words. These words act as ‘pivots’ whereby patterns are discovered using the presence of these words in various settings. The resulting 3996-word lexicon is reviewed by humans. Kim and Hovy [21] use a list of 38 positive and 40 negative seed words spanning the adjective, adverb and verb categories. Under the assumption that synonyms share the same polarity and antonyms have the opposite polarity, they use WordNet’s synonym and antonym word lists to augment their original seed list. They end up with thousands of positive and negative polarity words.

Hu and Liu [19] try to somewhat combine the two previous approaches. They first tag the documents with parts of speech tags and extract noun phrase patterns. If an adjective occurs in the verb phrase portion, they use it as part of the seed set and then look up synonyms for that adjective in WordNet. Blair-Goldensohn et al. [3] use a similar approach.

Thus far we have focused on approaches that use word lists known as lexicons. These lists usually consist of unigrams and bigrams that, on an intuitive level, would seem to be good indicators of sentiment. Other approaches strive for a deeper analysis of a document. These methods are often motivated by trying to model the difficult cases in SA.

Turney [35] and Mullen et al. [25] present similar methods of using extraction patterns based on parts of speech categories. The tagging is not only done at word

level (as in some of the other approaches mentioned) but rather the entire syntactic structures of sentences are taken into account. These extraction patterns are based on intuitive grammatical structures. Some relevant patterns can be seen if the first word is an adjective and the second word is a noun and vice-versa. If an adjective and a noun are tightly coupled in this way, it usually suggests that the adjective *modifies* the noun in a subjectively significant way.

Chesley et al. [8] attempt to combine the deeper linguistic analysis-based approach and the lexicon approach. They use a third-party application known as ‘InfoExtract’ and a Wikipedia-based dictionary to analyze text and sort appropriate words such as adjectives and verbs into synonym categories. They augment these features with various sentence and lexical *cues* such as where particular words appear in relation to the document, sentence and punctuation.

Finally, Whitelaw et al. [38] propose a deeper sentiment analysis scheme that uses *appraisal groups*. They aim to model such complex expressions as “not really very good” by combining the dimensions of these appraisal groups and enumerating scores that each group gives. Note that ‘orientation’ is only *one* out of *four* main dimensions. This is significant because other approaches *only* use orientation as a sentiment indicator. Also note that they use orientation to mean polarity (as the term is used in all other approaches), but use polarity to mean grammatical negation. In all, the appraisal groups can be thought of as *attributes* that subjective text is supposed to portray and they are defined as:

- Attitude - The emotional state and judgement of a situation
- Orientation - Positive and negative
- Graduation - How extreme is the judgement, the intensity

- Polarity - If there are negation modifiers such as ‘not’

2.4.2 Classification Methods

As we have seen, there are a variety of ways that seemingly salient features can be discovered from a document to indicate the document’s polarity. Some approaches even use a variety of different features such as unigrams, bigrams, parts of speech and syntactic and lexical structures. Having discovered these features, the second task is then to combine them in a way that will maximize the likelihood of a document belonging to a certain category over the other categories in question. For SA, this means that the goal is to increase the odds of the document being either positive or negative using the discovered features and to a lesser extent, analyze which particular features perform the best and under what circumstances.

The first set of approaches try to enumerate the features in a way that their *weighted* sum would fit the document into a polar category. For example, each feature that might indicate positive sentiment would be given a positive weight and a negative feature would be endowed a negative weight. It is how these weights are calculated and combined that distinguish the first set of approaches. Kim and Hovy [21] discuss two models to assign a sentiment category to a given sentence. The first model simply assigns a ‘+1’ weight to positive words and ‘-1’ to negative words and adds them up. If the total is positive, then the document is positively oriented. The second model also assigns the same weights but multiplies them instead. This model is concerned with the *presence* of words rather than the *frequency*, and according to the experiments that were carried out, the former performs better. The results achieved seem somewhat counter-intuitive at first because the expectation is that a high frequency of features that clearly belong to one category versus the other would clearly delineate the document.

Turney’s approach [35] uses a more fine-grained feature weighting scheme. As in the previous approach, the polar orientation of a sentence is calculated by endowing a weight to each feature and adding them up. However, this approach uses a “Pointwise Mutual Information (PMI)” score which is calculated by the *distance* between a word and the seed words “excellent” and “poor.” This distance is measured by searching for the word using AltaVista’s “NEAR” operator [35], which constrains the search to documents where the word appears within ten words of a seed word. This is further extended by Turney and Littman [36].

Instead of using a search engine, which can gather noisy results, Hu and Liu [19] and Ding et al. [10] use a similar weighting method as PMI but instead calculate the distance with the *relative minimal path length* using WordNet between a word feature in question and a seed word.

Mullen and Collier [25] experiment with combinations of these approaches, bringing all the diverse features together and classifying with Support Vector Machines (SVM). They discover that using the scores from multiple sources gives a higher accuracy than calculating scores from a single source.

Mullen and Collier’s approach has also brought into the forefront the machine learning approaches for SA. A defining strategy used in the Machine Learning approaches is to treat the classification task of SA as a specialized case of general text classification. Essentially this amounts to selecting the salient features as discussed in the Feature Selection section (see 2.3) and utilizing the *supervised learning* algorithms introduced in the Text Classification section (see 2.1). Therefore, most of the approaches in 2.3 are still valid and can in fact be used as part of a supervised learning approach.

Pang et al. [30] were one of the first groups to use the ML approaches for SA. Their objective was to use general text classification techniques and apply them to the

SA domain and discover the ways in which features for sentiment and features for topical classification are necessarily different and why this is the case. To accomplish this task they combine various feature representations such as unigrams, bigrams and trigrams with various methods of feature weighting such as *frequency*-based and *presence*-based methods. These feature characteristics are then tested with three supervised learning algorithms (see 2.1), and the results tabulated.

Using *movie review* data that was extracted from a *Usenet* movie review forum and annotated by the research group (each review was given a positive or negative rating), they selected the most frequently occurring features by *document frequency* and compared three supervised learning approaches (NB in 2.1.1, ME in 2.1.2, as well as SVMs). To ensure a 50% random choice baseline in their results, they selected an equal number of positive and negative review documents.

To test the feature weighting methods, they compared the accuracy achieved by the classifier when using a unigram representation and separately using a bigram representation. Their experimental results suggest that the *presence*-based feature weighting scheme outperforms that *frequency*-based weighting scheme. This is one of the fundamental differences between SA and general text classification, which is further discussed in [29].

Their comparison of the feature representation methods suggests that unigrams are better than bigrams and trigrams. As for the supervised classification methods, they find that SVMs outperform the other approaches, but not by a statistically significant margin.

A surprising result from this line of inquiry is related to the comparison of the most frequently-occurring unigrams (by DF) and a list of adjectives. Intuitively, the expectation is that the adjectives would provide more salient information per feature

than simply the most frequently-occurring unigrams, which may contain nouns and other seemingly unimportant parts-of-speech types. However, their findings indicate that simply utilizing adjectives significantly *degrades* accuracy in a supervised classification approach.

As we discussed in the Feature Selection section (see 2.4.1), adjectives are certainly not the only word types that can convey sentiments. Indeed adverbs, verbs and even certain nouns can convey subjectivity, so the approach taken by Pang et al. in terms of adjective features can be improved upon. Chesley et al. [8] use a much wider range of features and combine them in ways to attempt to discover the best combination. In addition to using basic adjectives and adverbs as their parts-of-speech features, they focus many of their experiments to subjectively-inclined verbs such ‘praising’, ‘approving’, and ‘doubting,’ identified by the InfoExtract system and the Wikipedia dictionary. To a lesser extent, they also include experiments with textual features such as question marks and exclamation points. These features are combined together into a feature set and the resulting polarity is classified using an SVM classifier. Given a collection of manually labelled blog posts and news articles from various web forums, they extracted 1076 documents for training and 76 for testing, both of which are divided evenly between positive and negative orientations. Using all the textual and linguistic features available, their classifier correctly identifies 84.2% of positive documents, 80.3% of negative documents, and 72.4% that contain no polarity (i.e., they are *objective*).

So far, the machine learning approaches described have mainly focused on bag-of-word features. Whitelaw et al. [38] combine simple bag-of-word features with appraisal groups (see 2.4.1) to achieve good accuracy. Using the same movie review data as [30], they build a base lexicon and augment it with a more detailed semantic taxonomy of appraisal groups (defined as “coherent groups of words that express together a particular attitude such as ‘extremely boring’, or ‘not very good’ ”). These features are different

from the standard n-grams because they have semantic meanings. Thus, each word could potentially be replaced by another semantically-similar word and the feature would still apply. Combining these features and using a variant of SVM, Whitelaw et al. achieve a 90.2% accuracy, which is the current state of the art accuracy level.

2.4.3 Topic Modelling Approaches to SA

The topic modelling techniques described in section 2.2 have been previously used as part of an overall sentiment analysis system. This is especially true for the LDA model because it is very modular and as such, can be readily extended for a wide variety of tasks in many problem domains. However, most of these approaches utilize a topic model as a way to extract *aspects*, which are the *subjects* of a review (i.e., the person or thing that is being rated). This is due to the fact that topic models (and LDA in particular) treat each document as a mixture of topics. The topics with the highest probabilities in a document then get mapped into the aspect space.

One such approach can be seen in Titov and McDonald [34]. Attempting to model online reviews on hotels, they posit that modelling aspects based purely on LDA is insufficient because they tend to mix *local* aspects such as ‘atmosphere’ and *global* aspects such as ‘London’. They create their own model, MG-LDA (Multi-Grain LDA), that gives a separate probability to local and global aspects so that their mixture can ultimately be modelled. After extraction, they do a set of qualitative and quantitative experiments, looking at the aspects that they extracted to see if they are good enough and using an aspect ranking system called *PRank* to compare the aspect rankings of their system with a gold standard. Their experiments on hotel reviews and electronics products suggest that their approach beats the standard LDA.

However, their approach is simply to extract product reviews and not to classify

them based on polarity. Their follow-up research on the other hand [3] uses the MG-LDA model to extract the most relevant aspects and attempts to classify their polarity on a *scale*. The first step of their hybrid model is to construct a general sentiment lexicon (similar to Hu and Liu [19] with a few additions to the system) that have a positive, negative and neutral probability for each word. Then for each aspect that is extracted, they calculate the sum of the weights for each word (representing probabilities of belonging to a polar category). If this sum falls into a pre-defined range over categories, then that aspect is classified appropriately. Their reported precision and recall rates are competitive with other aspect-based methods, although an *overall* sentiment polarity score (for the entire review) is never reported.

In a recent study by Lin et al. [23], the sentiment polarity is built in to the LDA topic model, so that their described model, the Joint Sentiment-Topic Model (JST) has a distribution over the sentiment for the entire document. Their generative model is similar to the LDA generative model (see subsection 2.2.2) in that a topic z is chosen from a topic mixture distribution and a word w is chosen from the distribution of words defined by topic z . The JST model adds another parameter l , the sentiment label, which is modelled by a sentiment mixture distribution for each document over all sentiment labels (positive and negative). Thus, for each document, a topic z is generated and a sentiment label l is chosen. The word w is then chosen by indexing a *joint* sentiment/topic-document distribution and a joint sentiment/topic-word distribution. After inference, it is the sentiment/topic-document distribution that determines the overall polarity of the document. It is important to note that although the authors claim that this is fully *unsupervised* approach, where the sentiment gets learned automatically, the prior sentiment-word distribution holds the mappings of words by sentiment category, which itself was defined by a lexicon [19]. Their experiments on the Pang et al. [30] movie review

data show competitive accuracy results, although somewhat lower than the current state-of-the-art approaches

Topic models do not necessarily have to be restricted to be unsupervised models. The supervised LDA model by Blei and McAuliffe [4] extend LDA to include a distribution of documents over pre-defined categories with the idea that the topic mixture within a document would indicate a probability distribution over the categories. In *supervised* LDA, the distribution of documents over categories is indexed by a *response* variable γ . This response variable is different from the topic variable z because γ is not restricted in indexing discrete values, but can be used to index real values and other types. As such, the response variable is chosen from a *Gaussian* distribution and it can therefore be thought of as a linear *regression* of *empirical* topic frequencies with Gaussian noise. Empirical in this case means that the document (words and topic assignments) is generated first and based on these topic frequencies, the response variable is generated. To model discrete types, or when the response variable cannot be sufficiently transformed to a Gaussian, the authors replace the linear model with a *generalized linear model*. The resulting topics in this model are proportioned not only to a ‘semantic’ word similarity defined in regular LDA but also to the overall linear model given by the response variable. Their experiments on *scaled* movie review data [28] show competitive results.

2.4.4 Discussion

In this chapter, we have explored a range of techniques for extracting some *meaning* from text documents and further applying various algorithms to make classification decisions based on those semantic relationships. Furthermore, we have outlined the formal mathematical derivations for each method which enables us to readily combine them into a solution for feature selection of sentiment analysis. In particular, by using

topic and syntax models in conjunction with supervised text classification techniques, we can approach the task of separating the semantic words from the syntactic words and classifying the polarity based on the syntactic features.

Chapter 3

Proposed Feature Selection Methods for Sentiment Analysis

As discussed in section 2.4, Sentiment Analysis can be divided into two stages, akin to general text classification. Accordingly, our system is also divided between these two tasks. For the first task, known as *Feature Selection*, we rely on the HMM-LDA (see subsection 2.2.6) topic model to extract the *salient features*. For the second task, known as *Document Classification*, we use a *Maximum Entropy* classifier (see subsection 2.1.2) to classify a review document into a positive or negative sentiment.

3.1 Feature Selection (FS) for Sentiment Analysis

3.1.1 Problems with Lexicon-Based Approaches

In Chapter 2, we discussed that for any automatic classification system, it is of the utmost importance to convert a piece of text into salient features. There are differing views as to what constitutes a salient feature, depending on the specific task at hand, the methods involved, and the domain. Because SA involves features that are associated with subjectivity, affect, emotion, and points-of-view, it is more difficult to justify using some features over others. This is at odds with selecting features for general text classification, where the task is to classify documents based on categories that are

relevant to the *frequencies* of the words contained in the documents.

Given the apparent differences between SA and general text classification, many researchers have used the *lexicon*-based approaches [3, 9, 10, 19, 20, 23, 43]. This line of thinking appears to be intuitive as human reviewers tend to use the same words to describe different subjects [29].

However, lexicon-based approaches have several key difficulties:

- *They take time to compile.*

In the process of building a list of words that could potentially hold some sentiment information, researchers often need to think about three different dimensions: What is the polarity of the word? What is the strength of the polarity? Is the word common enough? Considering each word in the lexicon on all decisions is guaranteed to take a lot of time, but researchers are often reticent in reporting the actual person-hours needed to analyze and compile a list of key sentiment-bearing words. One exception that is reported in Whitelaw et al. [38], is that their feature selection process took 20 person-hours.

- *They are based on human annotators.*

Deciding on what exactly sentiment is can be a subjective task in itself. Words such as ‘poor’ and ‘good’ have a clear polarity. However, words such as ‘still’ and ‘slow’ can have different interpretations depending on the *context*. In separate qualitative experiments done by Pang et al. [30], Wilson [40], and Kim and Hovy [21], the agreement between human judges when given a list of sentiment-bearing words is as low as 58% and no higher than 76%. This suggests that a lexicon-based approach cannot be a *general* solution to SA because it is based on the subjective judgements of salient features.

- *They are usually domain-dependent.*

Building a lexicon of words that can be used across domains is an impossible task. An example given by Turney [35] is the word ‘unpredictable’: It can be positive in a movie review, but negative in an automotive review. Hurst and Nigam [20] mention that their lexicon is manually tuned to the topic of their data. So is Das and Chen’s approach [9]. Given that SA is an inherently domain-dependent task, these approaches suggest that manual tuning is necessary for each new domain.

- *They may not be frequent enough to be useful*

When defining a lexicon, it is usually not clear which words hold the most relevance and importance. Researchers [9,21,40] usually end up with many *thousands* of words that they use as relevant features. Even though the English language has many thousands of words that indicate sentiment, the vast majority of these words are rarely used in a particular corpus [29]. In various machine learning approaches that use a lexicon, the final number of features becomes important for both computational efficiency and classification relevancy. Pang et al. [30] compared different *sizes* of features and have discovered that a list of only 2633 features is more accurate than any other list of greater size. Therefore, it seems that in machine learning approaches, having more features does not produce better accuracy, and in fact, may even diminish it.

In Chapter 2 we also discussed a number of strategies to select relevant features based on a fine-grained analysis [8,25,35,38]. These approaches are motivated by the desire to handle some of the most challenging cases in SA. In general, these approaches achieve a higher accuracy when classifying sentiment polarity, but also raise a few concerns, namely:

- *Generalizability*

Syntactic and linguistic analyses rely on finding *patterns* in the text that can be used as *clues* for suggesting sentiment polarities. Approaches such as Turney [35] and Mullen et al. [25] use POS tags and their order and distances to form such patterns and rules to extract relevant phrases. However, using grammatical structures is at odds with a review’s inherent *misuse* of grammar [29]. It could be the case that *most of the time* a syntactic pattern is useful, but due to the variations of an author’s writing style and language ambiguities, it could fail more frequently than a model that is not based on such restrictive rules.

- *Manually-developed extraction patterns*

Mirroring the approaches of manually compiling a lexicon, developing extraction patterns is also a painstaking task [29]. As natural language has countless ambiguities and exceptions to rules, it is difficult to justify manually developing *patterns* using empirical evidence. This can also be readily seen from an example. The statement “X beats Y” can be seen as positively orienting ‘X’ or negatively orienting ‘Y’. However, no information is given on whether ‘Y’ is indeed negative. Y could be positive, and it is just the case that X is *better* [29]. Thus, the most appropriate action is to consider the word ‘beats’ as having a positive orientation.

- *Feature Frequency*

Another concern when attempting to model the ‘edge’ cases in SA is how to model the weights of all these features. Since the edge cases do not occur often, or even not at all, the relevancy distribution of these features must necessarily be lower than the cases which occur more often; otherwise, the overall system would *inconsistently* categorize the polarities.

- *Domain dependency*

Reviewers tend to use different syntax when describing entities in different domains. For a movie review, the sentence “This actor is awesome” makes more sense than if a similar sentence such as “This stock price is awesome” is used in a financial review. Not only is the word ‘awesome’ ambiguous when it comes to stock prices (awesome as in ‘high’ or awesome as in ‘good value’), but financial reviewers also necessarily need to use more specific syntax to describe stock prices. It is for these reasons that Das and Chen [9] recommend alterations when using their finance-based SA system for other domains.

It is due to the problems discussed above that we aim to use a *shallow* machine learning approach which automatically selects the features that are both salient (at the word level) and important in terms of their frequencies, their positions, and the domain at hand.

3.1.2 Characteristics of Salient Features

To motivate our approach, we first describe criteria that are useful for selecting salient features for SA (see subsection 1.5). We describe each criterion along with some discussion. By salient features, we mean that:

1. *Feature should be expressive enough to add useful information to the classification process.*

As discussed in Chapter 2, the most expressive features in terms of polarity are *modifying* words that describe an entity in a certain way [14, 15, 25, 27, 30, 31, 35, 38, 39]. Some of these words include:

- Adjectives that describe an entity being discussed. For example, *good*, *bad*, and *ugly*.
 - Adverbs such as *terribly*, *awfully*, *amazingly*
 - Superlative words that describe an aspect in terms of another or describe an aspect in terms of some idea. For example, *better*, *best*, *worse*, and *worst*.
 - Nouns or verbs that relate an aspect in terms of a conscious ideal. For example, *love*, *hate*, *gem*, and *crap*.
2. *All features together should form a broad and comprehensive viewpoint of the entire corpus.*

In a corpus of many documents, some features can represent a subset of the corpus very accurately, while other features represent another subset of the corpus. The problem arises when representing the whole corpus with a specific feature set [32]. Therefore, all the selected features should be able to represent each document with a satisfactory level of accuracy (above a certain threshold). This can be readily tested using a *cross-validation* evaluation scheme.

3. *Features should be domain-dependent.*

One of the examples that is constantly used throughout the literature is how the term, *unpredictable* [35] is a good feature for movie reviews, but a bad feature for automotive reviews. Similar examples can be seen in [9, 20] as well. Since SA is a domain-dependent task, the final features should reflect the domain of the corpus that they are representing. Clearly, a *supervised* machine learning approach that picks up on such important cues from the domain is desirable.

4. *Features should be frequent enough.*

When selecting a list of features, it is important to take into consideration how often these features are used in the corpus. Machine learning approaches specifically operate on the frequencies of the selected features to determine the category of a document (see 2.1). If the features do not occur with a relevant frequency, the machine learning algorithm will find it difficult to learn based on them. In the experiments by Pang et al. [30], different feature set sizes were compared as well as the feature sets that had the same size but different constituents. Their results indicate that having more features does not help learning and in fact, the best accuracy was achieved by selecting features based on their *document frequencies*. This feature set also performed better than a more *informative* feature set composed of only adjectives. Their conclusion states that even though the adjectives are more relevant, most of them are not frequent enough in the corpus to be usable for machine learning.

5. *Features should be discriminative enough. A machine learning system needs to be able to pick up on their presence in certain documents for one outcome and absence in other documents for another outcome in classification.*

This relates to the general problem of feature selection in that each feature has to be relevant to the mapping of a document to a category. In the case of general text classification [32] the frequency of a feature in a document suggests that there is a relationship between the document and that feature. Features with high frequencies indicate a greater probability of the document having a strong relationship with these features. Features with low frequencies indicate the opposite. In the context of SA, this relationship does not readily apply [30], thus each feature should have inherently discriminant characteristics. Therefore, the task of identifying documents

based on a polarity should have as its feature set as many words as possible that are expressive and discriminative, and as few words as possible that have a *neutral* sentiment.

To obtain the features that satisfy the conditions above, we need methods that:

1. *Are based on probabilities.*

Having a probability-based system for feature selection is advantageous because the information endowed to each feature is based on how the particular feature *interacts* with the document (locally) and to an extent, the entire corpus (globally).

2. *Require minimal manual annotation.*

A system that does not require any human involvement is desirable over systems that do. This is because even though humans have an innate understanding of certain ideas and concepts and using this understanding can often make decisions that are better than an automatic system, they are often biased. As can be seen in various approaches [21,30,40] for SA, human annotators have a lower-than-expected agreement rate, and it is therefore better to have a system based on statistics.

3. *Work on any domain.*

We have seen how having features that perform well for a particular domain is desirable in a general SA system. This means that a *supervised* classification approach is important, since the classifier can learn based on specific, domain-relevant features. However, the actual system as a whole should be able to perform on any domain without having to apply any adjustments, as long as the necessary training examples are available for a particular domain. Systems that rely on manually-crafted rules [25,35] are at a disadvantage when there is a domain change because reviewers

will often use different wording and syntactic structure when reviewing different entities [29].

4. *Are computationally inexpensive.*

Although secondary in the theoretical development of any SA system, computational efficiency is nonetheless important. All things being equal, having a system that performs faster than another system is obviously an advantage. Looking at it another way, a system that is 100% accurate but involves exponential calculations or does not scale up linearly is certainly not usable in practise.

3.1.3 FS Based on HMM-LDA

Our proposed FS schemes use HMM-LDA (described in 2.2.6) to obtain words that, for the most part, follow the intuitive guidelines we set out in section 3.1.

At the heart of the HMM-LDA approach is the separation of *content* and *syntax*. Briefly, HMM-LDA is a composite model that models two kinds of *dependencies*: Long-range dependencies, which are modelled by the LDA topic model, utilize unique topic mixtures per document so that the content words can be found across documents (only in different proportions). Short-range dependencies are handled by the HMM component which is focused on finding word *patterns* in sentences and phrases, known as the syntactic structures.

In our discussion on feature selection, and indeed, as can be seen in other approaches, the salient features for SA do not typically involve words relating to the *subjects* and *objects* of the document (the *entities* that are being reviewed). This is why general text classification features are found consistently lacking in SA tasks.

For example, taking a segment from a randomly chosen document¹ in Pang et

¹The example was annotated by an editor independently

al 's movie review corpus [30], we can see how entities and modifiers are related to each other

Of course , it helps that **Kaye** has an **actor** as *talented* as **Norton** to play **this part** . It's *astonishing* how *frightening* **Norton** looks with a shaved head and a swastika on his chest . in addition to getting the look just right, **he's** *perfect* for this role – **Derek** requires intelligence, depth , and a whole lot of shouting , and **Norton** does it all with ease . Even when **he's** at his meanest, **Derek** has a *likable* quality to him, and that's a *gutsy* approach when telling a story about a skinhead . What adds depth to the story is a subplot in which **the principal of Danny's school** (**Avery Brooks**) becomes obsessed with purging the hatred from **Danny** . The other **performances** are all *terrific*, with standouts from **Furlong**, **D'Angelo**, and **Lien** . Visually, **the film** is very *powerful* . **Kaye** indulges in a lot of *interesting* **artistic choices**, and most of **them** *work nicely* –

Indeed, most of the information about an entity that relates it to a particular polarity comes from the *modifying* words . In the example above, these words are adjectives such as *talented*, *frightening*, *perfect*, *likable*, *terrific* and *powerful* . They can also be verbs such as *work* and adverbs such as *nicely* . The entities are represented by various nouns and pronouns such as *Kaye*, *Norton*, *actor*, and *them* .

Generalizing a single entity to all the entities that are found in a document, and taking all their respective modifiers into account, we can start to infer the polarity of the entire document based on the set of all the modifiers . This reduces to finding all the syntactic words in the document and disregarding the entities .

Separating content from syntax words in this way leads us naturally to a congruency with the lexicon-based approaches regarding the resulting features . An important distinction, however, is that these words were discovered automatically by HMM-LDA while the lexicon-based approaches require at least some human involvement . There are other advantages over lexicon-based approaches that will be discussed, but for now, a more detailed description of the framework is needed .

3.1.4 HMM-LDA Training Procedure

Figure 3.1: Diagram of the parsing and HMM-LDA training procedure.

As shown in Figure 3.1, the HMM-LDA training procedure is done in two stages.

- WS is a $1 \times N$ vector where WS_k contains the vocabulary index of the k th word token, and N is the number of word tokens. The word indices are not zero based, i.e., $\min(WS) = 1$ and $\max(WS) = |V|$, which is the number of distinct words in vocabulary V . A word index of 0 denotes the end-of-sentence marker. This marker is any one of a set of characters delineating a sentence such as '.', '?', '!', '.'. Note that the words are ordered according to their occurrences in documents.

- DS is a $1 \times N$ vector where DS_k contains the document index of the k th word token. The document indices are not zero based, i.e., $\min(DS) = 1$ and $\max(DS) = D$, which is the number of documents.
- V is a $1 \times |V|$ vector of word types where v_k contains the k th vocabulary item and $|V|$ is the number of distinct vocabulary items. The Gibbs sampler does not actually need the vocabulary, only the WS and DS vectors. However, the vocabulary is necessary when writing out the topics and classes.

The WS and DS vectors are given to the HMM-LDA Gibbs sampler as inputs. The Gibbs sampling procedure considers each word and document index, and estimates the probability of assigning the current word token to each topic, conditioned on the topic assignments to all other word tokens. From this conditional distribution, a topic is sampled and stored as the new topic assignment for this word token. Similarly, for each word index, the sampler estimates the probability of assigning the current word token to each class, conditioned on the class assignment of all the other words.

There are two phases in the Gibbs sampling procedure. The *initialization* phase and the *sampling* phase.

In the initialization phase, the sampler initializes the topic and class assignments for each word w_i by randomly assigning a topic and a class. We can represent these assignments by two vectors $\mathbf{z} = \{z_1, \dots, z_n\}$, a sequence of topic assignments where each z_i is one of K topics, and $\mathbf{c} = \{c_1, \dots, c_n\}$, a sequence of class assignments where each c_i is one of C classes. One class, $c_i = 1$ is designated as the ‘semantic class’, and the rest, the ‘syntactic’ classes.

In the sampling phase, for each sampling index i , where $0 \leq i \leq n$, we sample a topic from the conditional distribution $P(z_i | \mathbf{z}_{-i}, \mathbf{c}, w)$ and a class from the conditional

distribution $P(c_i | c_{-i}, z, w)$ (see section 2.2.7). In order to sample the variables z_i and c_i conditioned on all the other variables, the sampler keeps track of the assignment counts using 3 matrices:

- **WP** is a matrix of size $|V| \times |K|$ such that $WP_{i,j}$ contains the number of times word i has been assigned to topic j .
- **DP** is a $|D| \times |K|$ matrix, where $|D|$ is the number of documents, and $DP_{i,j}$ is the number of times topic i has been assigned to document j .
- **MP** is a $|W| \times |C|$ matrix, where $MP_{i,j}$ contains the number of times word i has been assigned to class j . Note that HMM state 1 represents the LDA model and $2 \dots |C|$ represent the “syntactic” HMM states.

Therefore, the sampler calculates each variable from the distribution of that variable conditioned on all other variables by *decrementing* the current assignment, calculating the conditional distribution, and then *incrementing* the count matrices with the new assignment. This procedure is outlined in Algorithm 3.

Algorithm 3 returns the count matrices WP , DP and MP as well as the topic assignments $\mathbf{z}$ and class assignments $\mathbf{c}$. For our approach, we do not need the topic assignments as we are dealing exclusively with the class assignments. As such, to estimate the conditional probability of word w_i being assigned to class c_j , we simply evaluate equation 3.1.

$$P_{c_j}(w_i) = \frac{MP_{w_i, c_j} + \beta}{\sum_k MP_{k, c_j} + W\beta} \quad (3.1)$$

where β is a pre-selected constant that represents the topic-word prior.

Algorithm 3 HMM-LDA Gibbs Sampler

```

{Initialization phase}
initialize all count matrices  $WP$ ,  $DP$  and  $MP$  to zero
for  $i = 1$  to  $n$  do
  sample class assignment  $c_i \sim \text{Multinomial}(\delta)$ 
  sample topic assignment  $z_i \sim \text{Multinomial}(\beta)$ 
  if  $c_i = 1$  then
    increment document-topic count:  $DP_{z_i, d_i}++$ 
    increment topic-word count:  $WP_{w_i, z_i}++$ 
  else
    increment word-class count:  $MP_{w_i, c_i}++$ 
  end if
end for
{Sampling phase}
for Number of Iterations do
  for  $i = 1$  to  $n$  do
    if  $c_i = 1$  then
      decrement document-topic count:  $DP_{z_i, d_i}--$ 
      decrement topic-word count:  $WP_{w_i, z_i}--$ 
      sample topic assignment  $z_i \sim \text{Eq. 2.30}$ 
      sample class assignment  $c_i \sim \text{Eq. 2.32}$ 
      increment document-topic count:  $DP_{z_i, d_i}++$ 
      increment topic-word count:  $WP_{w_i, z_i}++$ 
    else
      decrement word-class count:  $MP_{w_i, c_i}--$ 
      sample topic assignment  $z_i \sim \text{Eq. 2.30}$ 
      sample class assignment  $c_i \sim \text{Eq. 2.32}$ 
      increment word-class count:  $MP_{w_i, c_i}++$ 
    end if
  end for
end for

```

3.2 FS Based on Syntactic Classes

Before we can select the word features given by HMM-LDA, we must perform several pre-processing steps for each feature. Some punctuation marks can suggest sentiment (such as '!'), but in most cases they are irrelevant. Also, numbers and words that are combinations of letters and numbers can be safely removed as they most often indicate dates and various entities which are irrelevant to SA. Stop-words tend to appear very frequently in all contexts. They are not discriminative or informative enough to be of any use. The most common method to filter them out is to use a defined list of stop-words. Note that these steps could not have been performed before the training process because HMM-LDA depends on grammatical cues such as punctuation to model the syntactic structure of sentences. Moreover, stop-words are not removed prior to training because it might affect the transition probabilities of the syntactic classes.

Figure 3.2 HMM-LDA Feature Selection. 1 Semantic class and $|C| - 1$ syntactic classes are represented.

Figure 3.2 illustrates the process by which words in the semantic class and words

in the syntactic classes are merged together according to a feature selection *scheme*. These schemes are outlined in subsections 3.3.1, 3.3.2 and 3.4. Let $c_j \in C$ be a class assignment. Previously, we have defined $P_{c_j}(w_i)$ as the probability of word w_i in class c_j where $c_j = 1$ indicates the semantic class. Since each class (syntactic and semantic) is a probability distribution over all words, we need to select the words from these classes that offer a good *representation* of the class. The representative words in each class have a much higher probability than the rest of the words. Therefore, we can select the representative words by the *cumulative probability*. Specifically, we select the top percentage of the words in a class whereby the sum of their probabilities will be within some pre-defined range. This is necessary since there are many words in each class with low probabilities in which we are not interested [33]. The cumulative distribution function is defined as:

$$F_j(w_i) = \sum_{P_{c_j}(w) \geq P_{c_j}(w_i)} P_{c_j}(w) \quad (3.2)$$

Then, we can define the set of words in class c_j as:

$$W_{c_j} = \{w_i | F_j(w_i) \leq \eta\} \quad (3.3)$$

where η is a pre-defined cumulative probability value such that $0 \leq \eta \leq 1$. Next, we define the set of words in all the syntactic classes W_{syn} as:

$$W_{syn} = \{w_i | w_i \in W_{c_j} \text{ and } c_j \neq 1\} \quad (3.4)$$

and the set of words in the semantic class W_{sem} as follows:

$$W_{sem} = \{w_i | w_i \in W_{c_j} \text{ and } c_j = 1\} \quad (3.5)$$

Since modifying words for sentiment typically fall into syntactic classes, we could use words in W_{syn} as features for SA. However, as observed by Pang et al. [30], the best classification performance is achieved by a subset of features (typically around 2500). As a general step, we apply a document frequency (DF) cutoff to select the most frequent features. Let $df(w_i)$ denote the document frequency of word w_i , indicating the number of documents in which w_i occurs in the corpus. Then, the resulting features selected based on df can be defined as:

$$cut(W_{syn}, \epsilon) = \{w_i | w_i \in W_{syn} \text{ and } df(w_i) \geq \epsilon\} \quad (3.6)$$

where ϵ is the minimum document frequency required for feature selection. Essentially, we are ranking the terms in W_{syn} by df , and then removing the words with rank that is lower than a specified value.

3.3 Semantic-Syntactic Overlap Reduction Techniques

The main characteristic of using HMM-LDA classes for feature selection is that the set of words in the syntactic classes and the set of words in the semantic class are not disjoint. In fact, there is quite a significant overlap. In this section and the next, we discuss ways that we can try to remedy and even exploit this situation to get a higher level of accuracy.

First of all, it is important to realize that without a cumulative probability cutoff where only the most probable words are selected in each class, the overlap between any two classes would be 100%. This is because each class is a *multinomial* probability distribution over *all* the words in the vocabulary. Once we have selected a subset of the most probable words from each class, we have significantly reduced the overlap. However, quite a large

overlap still remains between the *merged* sets of words in the syntactic classes and the semantic class. This can be explained by the fact that simply having more words (in the merged W_{syn}) will necessarily increase the chances of an overlap between W_{syn} and W_{sem} . Another reason is in how HMM-LDA assigns classes to words in the first place. Following the procedure outlined in Algorithm 3, we can see that a class label for a word w_i gets chosen by calculating a conditional probability based on all the other class assignments and the transition probabilities assigned in the HMM component. As a result, a typical, high-frequency word can be assigned to more than one class depending on its location in the structure of a sentence.

The methods that we outline next are forms of overlap reduction techniques between W_{sem} and W_{syn} that aim to reduce the influence of the non-sentiment-carrying neutral features. A simple way of looking at it is that we are *removing* these neutral features from W_{syn} so that we have space in our final feature set for more suitable sentiment-carrying features.

3.3.1 FS Based on Set Difference Between Syntactic and Semantic Classes

In our first systematic approach, we try to see if we can get better accuracy by lowering the ratio of semantic words in our final feature set. We do so by removing the overlapped words that occur in W_{sem} .

More formally, given the set of syntactic words W_{syn} , we can get rid of the overlap by doing a set difference operation:

$$W_{syn} - W_{sem} \tag{3.7}$$

This will give us all the words that are more favoured in the syntactic classes. However,

as we shall see shortly, and also as we earlier speculated, by subtracting all the words in the semantic class, we are actually getting rid of some useful features. This is because (a) it is possible for the semantic class to contain words that are syntactic, and as a result are useful, and (b) there exist some semantic words that are good indicators of polarity. Therefore, we seek to ‘lessen’ the influence of the semantic class by cutting only a certain portion of it out, but not all of them.

In Algorithm 4, we outline a process that selects features from W_{syn} by applying a percentage cutoff for W_{sem} and then doing a set difference operation. We define $top(W_{sem}, \delta)$ to be the $\delta\%$ of the words with top probabilities in W_{sem} .

Algorithm 4 Syntactic-Semantic Set difference

Require: W_{syn} and W_{sem} as input

- 1: $W'_{sem} = top(W_{sem}, \delta)$
 - 2: $W_{diff} = W_{syn} - W'_{sem}$
 - 3: $W'_{syn} = cut(W_{diff}, \epsilon)$
-

When doing a set difference operation between W_{syn} and a percentage subset of W_{sem} we are essentially keeping all the words in W_{syn} that do not exist in W_{sem} , and at the same time, removing the words in W_{syn} that *do* exist in W_{sem} . This dual operation insures that we are only keeping the words in W_{syn} that are truly syntactic words.

Finally, addressing a range of δ values enables us to experiment with different sizes of subsets of W_{sem} to remove from our final feature set. A boundary condition of $\delta = 1$ implies the same result as $W_{syn} - W_{sem}$, which means that we are removing *all* the words present in W_{sem} . The other boundary condition of $\delta = 0$ gives us the set of words in W_{syn} . In subsection 4.6.1 we experiment with a range of δ values between the two boundaries and attempt to experimentally determine at which point the overlap is sufficiently reduced to give us an optimal classification accuracy.

3.3.2 FS Based on Max Scores of Syntactic Features

The running theme through the HMM-LDA feature selection schemes is that if a word is highly ranked (has a high probability of occurring) in a syntactic class, we should use that word in our feature set. Moreover, if a word is highly ranked in the semantic class, we usually do not want to use that word in our feature set because the word usually indicates a frequent noun. Therefore, the desirable words are those that occur with high probabilities in the syntactic classes, but do not occur with high probabilities in the semantic class, or do not occur there at all.

To this end, we have formulated a scheme that adds such words to our feature set. For each word, we obtain its highest probability in the set of syntactic classes. Comparing this probability with the probability of the same word in the semantic class, we disregard the word if the probability in the semantic class is greater.

For example, the word ‘love’ is a high-frequency, salient feature that could be in both W_{syn} when it is used in conjunction with an entity to describe a positive feeling towards that entity or it could be in W_{sem} as someone’s last name. Given that the word ‘love’ is used more in the context to describe a positive feeling in a review than it is used as a last name, the Max Score algorithm will include it in the final feature set because the probability of it being in W_{syn} will be higher, even though it could also be quite high in W_{sem} . We define the max scores for word w_i for both the syntactic and semantic classes and describe how we select features based on the max scores in Algorithm 5.

Note that word w_i must at least be in W_{syn} to be included in the final feature set. In this algorithm, we go through the whole vocabulary set obtaining the Max Scores for each word, before cutting a certain percentage of the words by the df cutoff. Analogously, this algorithm can be framed as iteratively removing high-probability, overlapped

Algorithm 5 Max Scores of Syntactic Features

Require: $c_j \in C$ where $1 \leq j \leq |C|$

- 1: **for all** $w_i \in V$ **do**
 - 2: $S_{syn}(w_i) = \max_{c_j \neq 1} P_{c_j}(w_i)$
 - 3: $S_{sem}(w_i) = P_{c_1}(w_i)$
 - 4: $W_{max} = \{w_i | S_{syn}(w_i) > S_{sem}(w_i)\}$
 - 5: **end for**
 - 6: $W'_{syn} = cut(W_{max}, \epsilon)$
-

words from W_{sem} . Again, this ensures that the final feature set contains as many high probability features as possible from W_{syn} , while at the same time making sure not to remove potentially salient features from $W_{sem} \cap W_{syn}$.

3.4 Selecting Syntactic Features Based on POS Categories

The hidden Markov model in HMM-LDA assigns classes to words based on how many classes were chosen and their relative positions in a sentence. This means that some syntactic classes are not as informative as others. Essentially, we are looking for those syntactic classes that have the parts-of-speech that are most relevant in a polarity classification task. The best indicators of polarity are usually adjectives and adverbs, but we are certainly not restricted to those.

Since HMM-LDA does not necessarily assign words to classes based on parts-of-speech, we cannot simply include the classes that *seem* to hold words with certain associated POS tags. As discussed in section 3.1.2, sentiment indicators are typically adjectives and adverbs. However, HMM-LDA is not guaranteed to place all the adjectives in one class and all the adverbs in another since there is a high variability of syntactic structure of sentences. Therefore, we cannot *only* include the classes that have a high ratio of salient syntactic POS words, but we can frame our approach to ignore those classes that have a low ratio of salient syntactic POS words.

Algorithm 6 describes a procedure to restrict some of our HMM-LDA syntactic classes in favour of classes that contain a reasonable overlap with the words in the POS categories. For example, if a class has a low percentage of adjectives or adverbs, we can exclude it from consideration. The words which we are intuitively seeking are tagged with tags from POS'

Algorithm 6 Selecting a Subset of Syntactic Classes by POS Categories

Require: POS be the set of all available POS tags

Require: POS' be a subset of POS tags of interest

Require: $\mathbf{p} = \{p_1, \dots, p_n\}$ is a sequence of parts-of-speech tags where each tag p_i is one of POS tags

```

1  $C_1 = C_2 = C_3 = \dots = C_M = 0$  for all  $M$ 
2  $T_1 = T_2 = T_3 = \dots = T_M = 0$ 
3 for  $i = 1$  to  $N$  do
4    $T_{c_i} \leftarrow T_{c_i} + 1$ 
5   if  $p_i \in POS'$  then
6      $C_{c_i} \leftarrow C_{c_i} + 1$ 
7   end if
8 end for
9  $C' \leftarrow \{C_j | \frac{C_j}{T_j} \geq \mu\}$ 
10  $W_{syn} = \{w_i | w_i \in W_{c_j} \text{ and } c_j \neq 1\}$  for all  $c_j \in C'$ 

```

Algorithm 6 calculates the ratio of words in a class where each word has been assigned to a specific POS tag to all the words in that class. Thus, each class will have a percentage of words from POS' . We then take the classes that have a percentage of POS' tags higher than some specified value μ and include them in further feature selection schemes such as Algorithms 4 and 5. Note that this scheme is only used to reduce the number of syntactic classes and is not used for evaluating the semantic class.

3.5 Discussion

In section 3.1.2, we outlined several key characteristics that are important for selection of salient features for a polarity classification task. Next, we discuss how these

characteristics relate to our proposed schemes.

The short-range dependencies extracted into the HMM-LDA syntactic classes pick up on the syntactic structure in a sentence. In a typical sentence describing some entity, there are nouns, adjectives, verbs and adverbs. Using a typical POS-tagger, we can separate these word types into relevant POS categories, and then ignore the noun category. However, as we have discussed in section 3.1.3, this process is not sufficient. Unlike tagging words by POS tags, the HMM-LDA model allows for more fine-grained groups (which are not restricted to previously defined groups) that use probabilistic co-occurrence relationships between words appearing in a sentence and document together.

When combining these groups it is clear that the *semantic* class holds most of the noun types that relate to the entities being described, which is due to the long-range dependencies between words that make up the entities. For example, in a typical collection of reviews on a particular subject, each review discusses certain entities and the characteristics of these entities. These entities and their characteristics are usually common in the whole collection. The semantic class holds the words that comprise these entities and their characteristics, while the syntactic classes (short-range dependencies) hold the *modifying* words that describe each entity and its characteristics on a sentence level.

In our approach, we exploit this separation of semantic and syntactic classes by ignoring the words in the semantic class while focusing on the words in the syntactic classes to describe the polarity of a document (see 3.1.3). Our reasoning is that in a typical review document, we do not care about the entities and their characteristics as much as we care about the modifiers that describe them. It is by aggregating these modifiers that we can classify polarity at the document level.

To ensure that these modifiers are frequent enough to be useful features, we

again note that each syntactic class is also a distribution over the vocabulary. Therefore, the likelihood of each word occurring in a sentence and document is necessarily governed by the empirical frequency of words in topics and classes. This means that the top words in each syntactic class must occur frequently enough so that they can be explained by the distribution of the class.

In section 3.3 we outlined two algorithms to reduce the overlap of the representative words in each class, and specifically, between the merged syntactic set W_{syn} and the semantic class W_{sem} . The reasons for doing so are discussed in that section. Abstracting from the specific algorithms and techniques, we are mainly interested in features that carry sentiment and that are *discriminative* enough to indicate a certain polarity. Each of these techniques allow us to focus on these discriminative features by reducing the presence of the *neutral* or semantic features that carry no sentiment.

Since our approach involves an unsupervised training process to select features and a supervised classification process to test them on a corpus, it requires no manual annotation. It only requires certain parameters to be selected such that the resulting model is dependent on the selection of these parameters as it pertains to the number of topics and classes and the prior distributions of the words within them. However, our feature selection schemes are based on *merging* these classes and as a result are not sensitive to the change in these parameters.

Finally, having a system that is computationally inexpensive is an advantage. Our FS approach is based on a Gibbs sampling approximation of an intractable distribution 2.2.4. As such, it relies on an iterative method of sampling from conditional distributions. As we outline in Algorithm 3, and demonstrate in our Experiments chapter, this sampling approach iteratively samples topics and classes for each word in the corpus, for each iteration. Since the HMM classes have a tighter dependency than the

topic assignments, the running time of HMM-LDA depends on the number of classes chosen. Since this number is usually very small, the training process is not computationally expensive. The other part of our system involves training a Maximum Entropy model (see 2.1.2) where the parameter estimation relies on an iterative optimization technique whose running time is proportional to the number of features and outcomes. Since we are dealing with a low number of features, the overall process is not computationally expensive either.

Chapter 4

Experiments and Results

4.1 Data Set and Evaluation

To measure the effectiveness of our proposed feature selection schemes, we conducted experiments on the *movie review dataset*¹ given by Pang et al [27, 30]

This dataset consists of 1000 positive and 1000 negative labelled movie reviews where each document d has a sequence of N_d words, denoted as $d = \{w_1, w_2, w_3, \dots, w_{N_d}\}$ such that $d \in D$, the set of all documents in the corpus, and $w_i \in V$, the set of all words in the vocabulary. The two outcome labels relating the polarity of a document are defined as $L = \{neg, pos\}$. Therefore, each training example consists of a pair (d, o) such that $o \in L$.

4.1.1 Corpus Statistics

Table 4.1 summarizes the general statistics of the movie review corpus. As can be seen, the average word length and sentence length are in line with the expected characteristics of documents in the English language. However, the corpus does have a high document length variance. In fact, the shortest document has 19 words whereas the longest has 2,879 words. The Lexical Diversity score suggests that on average, there is not much word repetition, which is in line with the indication that reviewers tend to use

¹<http://www.cs.cornell.edu/people/pabo/movie-review-data/>

Corpus Totals		
Total number of word tokens	1583820	
Total number of word types	39764	
	Average	Std. Dev.
Word length	4.32	0.468
Sentence length	23.13	5.680
Document length	791.91	347.251
Lexical Diversity Score	1.79	0.445

Table 4.1: General statistics for the movie review corpus. *Average word length* is the total number of characters divided by the total number of words in the corpus; *average sentence length* is the total number of words divided by the total number of sentences; *average document length* is the total number of words divided by the number of documents; and the *Lexical Diversity score* is the average number of times each vocabulary item appears in a document. The table shows the averages and standard deviations for the related statistics.

a wider range of language to describe an opinion [29].

4.1.2 Evaluation Procedure

The evaluation procedure is a two-stage process consisting of Feature Selection and Document Classification, which were first introduced in section 2.4:

1. Feature Selection

In the first stage, the goal is to extract features. This procedure corresponds to identifying the parts of the document that will most likely infer whether that document is *positive* or *negative*, and can be further divided into two steps:

- (a) *Training the HMM-LDA model* (corresponding to 3.1).

To train the HMM-LDA model we use the Topic Modelling Toolbox², which is a collection of MATLAB functions that encode and train LDA models using Gibbs sampling. For more information on the input parameters and matrices that are returned, see section 3.1.4.

²http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm

- (b) *Obtaining features based on the feature selection schemes* (corresponding to 3.2).

We combine the words in the resultant classes from step (a) into features using the methods outlined in subsections 3.3.1, 3.3.2 and 3.4.

2. Document Classification

To classify the movie review corpus for polarities, we first test the baseline methods, which are described in the next section.

Next, we utilize the schemes from 3.2 on two feature selection approaches. The first approach is to use the feature selection schemes on *all* the available classes produced by HMM-LDA training. The second approach selects a subset of the syntactic classes using POS tags (see 3.4) and then repeats the feature selection schemes (3.3.1 and 3.3.2) for these selected classes.

4.1.3 Evaluation Metrics

Throughout these experiments, we are interested in the *classification accuracy*. This is evaluated simply by comparing the resultant class from the classifier and the actual class annotated by [30]. The number of matches is divided by the number of documents in the test set. Thus, given an *annotated* test set $d_{test_A} = \{(d_1, o_1), (d_2, o_2), \dots (d_S, o_S)\}$ and the classified set produced by the classifier, $d_{test_B} = \{(d_1, q_1), (d_2, q_2), \dots (d_S, q_S)\}$, we calculate the accuracy by:

$$\frac{\sum_{i=1}^S I(o_i = q_i)}{S} \quad (4.1)$$

where $I(o_i = q_i)$ is the *indicator* function such that $I(o_i = q_i) = 1$ when $o_i = q_i$ and $I(o_i = q_i) = 0$ otherwise.

4.2 Baseline Results

The purpose of the baseline experiments is to compare our feature selection schemes against established approaches using the same evaluation metrics and to quantify the significance of any improvements that we have achieved. One of the established approaches is the FS scheme based on DF described by Pang et al. [30] and briefly outlined in 2.4.1. By replicating their approach, we can *anchor* our own efforts in terms of an established FS scheme, which defines a *lower bound* and gives us some idea of the significance level that we can achieve. Following this approach, we outline three other baseline approaches and evaluate them in the same way. The first one extends Pang et al.’s method with more documents and a higher number of iterations. The purpose of reporting the results of this baseline is to gauge the accuracy by using the same approach, but with parameters that will be used for our own schemes. The last two baseline methods are given as simple FS schemes based on parts-of-speech that are meant to test the intuition of our own approach, as well as set a quantifiable standard.

4.2.1 Pang et al.’s FS Based on Document Frequency (DF)

For the first baseline, we try to replicate the results of Pang et al.’s [30] maximum entropy classification experiments. Their features were generated by extracting the most frequent unigrams in the movie review corpus by DF, and training a maximum entropy classifier on those features. They used the standard classification accuracy evaluation methods outlined in subsection 4.1.3.

Pang et al.’s feature selection scheme:

- Maximum Entropy classification trained with Improved Iterative Scaling (IIS) (see subsection 2.1.2).

- 1,400 documents (700 negative and 700 positive)
- 2633 unigram features selected by document frequency (DF) cutoff
- 3-fold cross validation
- 10 iterations of training
- Gaussian prior smoothing

*Highest reported classification accuracy: **0.81***

4.2.2 Extended FS Based on Pang et al.’s DF

For the next three baselines, and for our own approach, we use the same evaluation methods, but with different training parameters. We extend Pang et al.’s approach by using the full 2000 documents in the movie review corpus, training the maximum entropy model using different numbers of iterations and reducing the number of features from 2,633 to 2,500.

In our experiments, we randomize the documents and split the data into 1,800 documents for training and testing and 200 for *validation*. The validation set is used as a ‘feedback’ mechanism for model training. Typically, if we set the number of iterations to a value that is too low, the model might not be trained properly, resulting in *underfitting*. On the other hand, if the number of iterations is too high, we might be *overfitting* the model. This is why it is often appropriate to use a validation set whereby after a certain number of iterations, the classifier can test the currently trained model against it. If the classification accuracy increases, we let the model train for more iterations. If the accuracy decreases, we stop the training process. This method corresponds to the ‘eval’ results that we generate, meaning that the actual number of iterations per fold is variable.

For the 1,800 documents, we run a 3-fold cross validation procedure where we train on 1,200 documents and test on 600. We compare the resultant feature sets after each FS scheme using the OpenNLP³ Maximum Entropy Classifier. This approach will constitute as the basis for SA and will subsequently be used for comparison with our approaches.

- Maximum Entropy classification (IIS)
- 2000 documents (1000 negative, 1000 positive)
- 200 documents in eval set
- 2500 unigram features
- 3-fold cross validation
- no smoothing

The results for this method are reported in Table 4.2.

4.2.3 FS Based on Parts-of-Speech (POS)

In order to compare our HMM-LDA based schemes with a method that follows our FS characteristics (in regards to having features which are *expressive*) from section 3.1.2, we ran experiments to explore a basic POS-based FS scheme. For this approach, we use NLTK's⁴ Penn Treebank POS tagger⁵. First, we tagged the words in each document with POS tags and selected the most frequently-occurring unigrams that were not tagged as 'NN', 'NNP', 'NNS' or 'NNPS' (the 'noun' categories). Next, we tagged all the words and only selected the ones that were tagged as 'JJ*', 'RB*' and 'VB*' categories (the 'syntactic' categories). Lastly, for comparison with our DF approach, we selected the top

³<http://incubator.apache.org/opennlp/>

⁴<http://www.nltk.org>

⁵<http://www.cis.upenn.edu/treebank/>

2500 features by DF. The idea is to include as part of the feature set all the words that are not ‘semantically oriented’. Note that this FS scheme does not correspond to the scheme outlined in section 3.4.

- Maximum Entropy classification (IIS)
- 2000 documents (1000 negative, 1000 positive)
- 200 documents in eval set
- 2500 unigram features
- 3-fold cross validation
- POS (-NN*)
 - POS tags not included: ‘NN’, ‘NNP’, ‘NNS’, ‘NNPS’
- POS (JJ* + RB* + VB*)
 - POS tags included: ‘JJ*’, ‘RB*’ and ‘VB*’

4.2.4 Baseline Results

Iterations	DF Cutoff		POS (-NN*)		POS (JJ* + RB* + VB*)	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
10	0.821	7.9×10^{-3}	0.827	5.4×10^{-3}	0.811	8.4×10^{-3}
25	0.836	3.9×10^{-3}	0.831	2.4×10^{-3}	0.824	6.9×10^{-3}
eval	0.845	3.4×10^{-3}	0.848	2.3×10^{-3}	0.826	1.1×10^{-3}

Table 4.2: Baseline Results with a different number of iterations. Each column represents a feature selection method.

From Table 4.2, we see that the FS based on DF cutoff is in line with Pang et al.’s FS scheme at 10 iterations, given the standard deviation. Note that we are using

the ‘Extended’ DF scheme in this case, where the differences with Pang et al.’s scheme is noted in section 4.2.2. By varying the number of iterations and using the full set of review documents available, we have shown an improvement over Pang et al.’s scheme.

We can also see that our POS-based approaches do not show any improvement over the DF-based approach. For **POS (-NN*)**, the lack of improvement in accuracy can be explained by the fact that as we are removing ‘noun’ types, we are getting rid of some expressive features. This is related to the characteristics outlined in 3.1.2 and is further backed up by the last scheme, **POS (JJ* + RB* + VB*)**, which suggests that potentially salient features are not restricted to a few POS categories. Overall, we can conclude that a more fine-grained approach is necessary when extracting sentiment-carrying features.

4.3 HMM-LDA Training

We trained the HMM-LDA model on the 2000 document corpus of movie reviews (the training procedure is outlined in 3.1.3). Since the HMM-LDA model requires sentences to be outlined, we used the usual sentence markers (‘.’, ‘!’, ‘?’, ‘:’).

We then trained 3 Gibbs chains, taking the last sample after 100 more iterations from each chain after a burnin period of 1000 iterations. The training parameters were **T** = 50 topics, **S** = 20 classes, **ALPHA** = 1.0, **BETA** = 0.01 and **GAMMA** = 0.1.

We found that the burnin period of 1000 iterations was sufficient enough as we tracked the log-likelihood every 10 iterations. These are shown in Figures 4.1 and 4.2. We can see that after 1000 iterations, the 3 Gibbs chains are close to one another and are starting to converge to a singular log-likelihood value. In the figure, this is shown by solid lines. Since the 3 chains were all initialized randomly, this suggests that the training

process is not sensitive to initial conditions. Therefore, taking the last sample from any of the 3 chains is sufficient for selecting a model.

Figure 4.1: HMM-LDA Log-likelihood by the number of Gibbs iterations. The 3 dotted lines represent the “burn-in” iterations for each chain. The 3 solid lines after 1000 iterations represent the sampled chains.

After training, we have both the topic assignments $\mathbf{z}$ and the class assignments $\mathbf{c}$ from the trained model as well as the count matrices shown in section 3.1.3. We calculate the probabilities for each word given a class from these count matrices using the formulae given in 3.1. An example of the word-class relationships that are given by the HMM-LDA model after training can be seen in Table 4.3.

4.3.1 Qualitative Analysis - Example Annotated by Classes

We take the sample segment from the document we randomly selected in section 3.1.3 to demonstrate how the HMM-LDA classes are assigned to words. The words outlined in the boxes are the same words that were annotated in 3.1.3, but instead of ex-

john	so	time	his	up	very	director	including	first	named
james	because	characters	her	him	some	writer	than	chewing	old
michael	how	scenes	its	out	no	screenwriter	especially	sh	de
star	however	films	one	them	more	watching	nor	second	called
david	then	fact	every	more	too	saving	whose	quack	school
alien	even	people	some	me	so	meet	whom	fritter	whose
joe	though	years	any	well	any	starring	yet	novella	york
kevin	although	movies	no	her	pretty	having	without	comer	van
hollywood	why	life	another	off	his	seeing	because	spazio	looking
love	after	thing	each	us	such	making	except	arranges	than
he	is	than	about	good	between	one	not	here	film
doesn	are	is	like	first	about	most	have	again	movie
don	was	before	than	bad	like	out	just	mr	story
isn	has	so	after	new	more	all	been	later	character
didn	will	about	over	other	behind	much	see	ago	man
one	have	just	during	great	are	some	make	dr	way
wasn	does	not	is	big	involving	lot	get	itself	world
won	had	being	around	best	against	little	even	matter	audience
aren	were	rather	without	real	before	more	all	movies	year
let	seems	show	off	two	include	bit	really	else	time

Table 4.3: Top 10 highest-probability words extracted by the HMM-LDA for every class. The first column is the semantic class and the other columns are the syntactic classes.

Figure 4.2: HMM-LDA Log-likelihood by the number of Gibbs iterations (closeup)

tracting the entities and modifiers, we label each annotated word with its class assignment.

This gives us a preliminary idea of the validity of our approach.

... Of course , it helps that [Kaye]⁰ has an [actor]¹⁶ as [talented]¹⁴ as [Norton]⁰ to play this [part]⁴. It's astonishing how [frightening]¹⁴ [Norton]⁰ looks with a shaved head and a swastika on his chest. In addition to getting the look just right, [he]^{'s}¹³ [perfect]⁶ for this role – [Derek]⁰ requires intelligence, depth , and a whole lot of shouting , and [Norton]⁰ does it all with ease. Even when [he]^{'s}¹³ at his meanest, [Derek]⁰ has a [likable]¹⁴ quality to him, and that's a gutsy approach when telling a story about a skinhead. What adds depth to the story is a subplot in which the [principal]¹⁶ of Danny's school ([Avery Brooks]^{0,0}) becomes obsessed with purging the hatred from [Danny]⁰. The other [performances]² are all [terrific]¹⁴, with standouts from [Furlong]⁰, [D'Angelo]⁰, and [Lien]⁰. Visually, the [film]¹⁶ is very [powerful]¹⁴. [Kaye]⁰ indulges in a lot of [interesting]¹⁴ [artistic]¹⁴ choices, and most of [them]⁴ [work]¹¹ [nicely]⁴ –

The first thing to notice about the paragraph above is that all the names such as 'Kaye' and 'Norton' are in class 0 (the semantic class). The second thing to notice

is that some of the adjectives such as ‘talented’ and ‘frightening’ are in the same class. Another result worth mentioning is that the assigned classes do not necessarily follow parts-of-speech categories. This is in line with the expected premise that HMM-LDA is a more fine-grained approach, discussed in section 3.5. However, we note that in this example, some common nouns such as ‘actor’, ‘film’ and ‘principal’ are in their own class, and not necessarily in class 0. We defer discussion on this issue for the later sections.

4.4 Selecting Representative Words for Each Class

The running theme in our feature selection schemes is to select *representative* words from all the classes. This is because each class in HMM-LDA is a distribution over all the words in the vocabulary. For our purpose, we only select the words that carry *most* of the probability for that class such that there is not a large overlap between the classes in the final feature set. We use all the syntactic classes because each class picks up on a portion of the overall syntactic structure and there needs to be a broad coverage of all the classes in all documents as argued in subsection 3.1.2.

In this experiment we fix $\eta = 0.9$ to get the top words in each class having a cumulative probability of 0.9. At this level, there are 6,189 words. Representative words in the syntactic classes are merged into W_{syn} . Finally, we cut the number of words to 2500 by sorting them using df and removing the rest since according to Pang et al. [30], the best classification performance is achieved by a subset of features. This list of words is then used as features for the Maximum Entropy classifier. We again run the classifier for 10, 25 and ‘eval’ number of iterations in order to compare with the baseline results.

To test if this FS scheme’s higher accuracy is *statistically significant* over the baseline results in Table 4.2, we perform **one-tailed t-tests** at the 95% significance level.

Iterations	Mean	Std. Dev.
10	0.823	1.0×10^{-3}
25	0.839	1.3×10^{-3}
eval	0.863	5.8×10^{-3}

Table 4.4 Results for the HMM-LDA FS method at 10, 25 and eval iterations

In each case, we perform the tests on the ‘eval’ number of iterations using the cross-validation folds as the samples. The p-values are 0.0195 and 0.0282 when comparing the mean accuracy of HMM-LDA FS to DF Cutoff and POS (-NN*), respectively. This indicates that the accuracy achieved by the HMM-LDA FS scheme is significantly higher than the two baselines.

4.4.1 Analysis for the Syntactic Features

The mean accuracy results from Table 4.4 have established an improvement in classification accuracy over the baseline methods from Table 4.2. To explain the improvement in accuracy, we examine the *differences* between the top words from the baseline DF scheme and our scheme based on syntactic features. Since the baseline POS approaches do not offer a significant improvement over the DF baseline, we shall examine the differences between the DF approach and our FS method.

We begin by qualitatively examining the two feature sets. This is done by performing a set difference operation between the features from W_{syn} and the features from the baseline DF method, here referred to as W_{df} . Note that in each case the actual difference in words is 24.6%.

- Top 25 words from $cut(W_{df}, 2500) - cut(W_{syn}, 2500)$

‘other’, ‘will’, ‘plot’, ‘made’, ‘world’, ‘own’, ‘minutes’, ‘star’, ‘job’, ‘written’, ‘problem’,
‘james’, ‘michael’, ‘sequence’, ‘opening’, ‘ends’, ‘five’, ‘book’, ‘career’, ‘bring’, ‘co’,
‘laugh’, ‘robert’, ‘chance’, ‘peter’

- Top 25 words from $cut(W_{syn}, 2500) - cut(W_{df}, 2500)$.

‘featured’, ‘picked’, ‘carefully’, ‘suffer’, ‘strangely’, ‘variety’, ‘horribly’, ‘storm’, ‘ordi-
nary’, ‘degree’, ‘demands’, ‘appearances’, ‘western’, ‘personally’, ‘silver’, ‘handsome’,
‘gift’, ‘cry’, ‘throws’, ‘providing’, ‘notion’, ‘wonders’, ‘downright’, ‘troubled’, ‘accom-
plished’

From these two lists, we can see that at the 2500 word level, the words that are present in W_{syn} but absent in W_{df} are more *salient* as per the requirements set in 3.1.2. Names such as ‘james’, ‘michael’ and ‘robert’ are clearly entity words and are not present in W_{syn} . Moreover, sentiment-carrying words such as ‘troubled’ and ‘accomplished’ are not in W_{df} . To quantify the improvement in classification accuracy, we discuss the relationship between the DF scores for each word and its probability in a class.

4.4.2 Correlation between Document Frequencies and Probabilities

Tables 4.3a and 4.3b show the top 25 words in classes W_{sem} and W_{14} , respectively. As can be seen, words with high probabilities also tend to have high document frequencies, which can be more readily seen in Table 4.3b. However it is difficult to assess this relationship by examining only the top 25 words in each case. Figures 4.4 and 4.5 illustrate this correlation for a higher number of words by overlaying the word probabilities for classes 0 and 14 with the document frequencies for each word.

w	$df(w)$	$P(w)$	w	$df(w)$	$P(w)$
john	470	0.005044	good	1182	0.019914
james	298	0.002714	first	1047	0.01335
michael	272	0.002638	bad	773	0.011740
star	460	0.002632	new	769	0.010478
david	264	0.002329	other	1108	0.01013
alien	130	0.002145	great	697	0.010000
joe	125	0.002032	big	654	0.0090918
kevin	169	0.002032	best	854	0.009015
hollywood	390	0.00200	real	623	0.0080590
love	656	0.00198	two	1062	0.0077248
jack	143	0.001879	few	697	0.0074666
robert	195	0.001835	same	632	0.007390
war	181	0.001803	only	1309	0.0071224
jackie	63	0.001792	little	921	0.0071031
disney	93	0.001791	own	593	0.0062526
black	297	0.001790	old	602	0.0062526
tom	157	0.001709	action	572	0.0062050
school	245	0.001689	young	508	0.0057463
will	1071	0.001651	long	637	0.0055640
peter	188	0.001651	funny	505	0.0053159
fiction	139	0.001626	last	601	0.0051630
lee	151	0.001607	special	367	0.0050284
earth	184	0.001601	interesting	482	0.0048471
scream	91	0.001575	original	421	0.0047803
played	550	0.001563	small	342	0.0040917

(a) W_{sem} (b) W_{14}

Figure 4.3: The top 25 words in W_{sem} and W_{14} sorted by the probability. The second column is the document frequency. The third column is the probability.

From Figures 4.4 and 4.5, we see that even though the probabilities and document frequencies for each word seem to be correlated, some words that do not have a high probability nonetheless have a high document frequency. This can be explained by the fact that HMM-LDA places a higher priority on modelling the short-range syntactic relationships in a sentence than the frequency of a particular word. This is in line with the explanations given in subsections 2.2.6 and 3.1.4.

As a final step in all our feature selection schemes, we always apply a DF cutoff to cut the number of features into a manageable size. This is only reasonable if the word

Figure 4.4: The bars represent the document frequency of each word. The line represents the word probability.

probabilities are correlated with their document frequencies. However, they should not be *highly* correlated since we would then get the same feature set as in the simple schemes based on DF after the cutoff. To justify these claims, we calculate the Pearson correlation for each class between the document frequencies and word probabilities for words chosen at $\eta = 0.9$ cutoff.

The Pearson correlation coefficient measures the linear relationship between two datasets, between -1 and +1. Correlations of -1 or +1 imply an exact linear relationship. Positive correlations imply that as x increases, so does y . Negative correlations imply that as x increases, y decreases. Correlations of 0 indicate no correlation. The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets.

From Table 4.5, we can see that word probabilities and document frequencies are correlated since every class has a positive correlation in the range of $[0.313938, 0.888160]$ where the average correlation is 0.575612.

Figure 4.5 The bars represent the document frequency of each word. The line represents the word probability.

Class	Pearson Correlation Coefficient	P-value
0	0.313938	0.0
1	0.540868	4.113330e-59
2	0.741573	0.0
3	0.556147	2.263138e-41
4	0.644826	0.0
5	0.639375	2.744460e-109
6	0.378960	6.813323e-22
7	0.466660	1.718667e-27
8	0.888160	3.836504e-17
9	0.469479	1.775012e-32
10	0.414316	4.018825e-14
11	0.511993	1.565403e-143
12	0.641896	1.544255e-62
13	0.695944	1.573257e-125
14	0.680440	0.0
15	0.439242	4.747465e-21
16	0.666601	8.220091e-196
17	0.660328	2.268749e-320
18	0.565037	8.223818e-98
19	0.596475	4.373702e-306

Table 4.5 Pearson correlation between word probability and document frequency per class.

4.5 Overlap Analysis

In the previous section we are concerned with selecting *representative* words from each class whereby we target the words that give 90% probability density to their respective classes. Each class is a multinomial distribution over the vocabulary, and after merging all syntactic classes into W_{syn} we obtain features of a broad coverage over the entire syntactic structure. In this section, we examine the *word overlap* between the syntactic classes before merging into W_{syn} and the overlap between the W_{syn} and W_{sem} . We first look at the the overlap between the syntactic classes.

After merging, $|W_{syn}| = 6,189$ words. We can calculate the ratio of unique words by dividing the number of words in the merged set W_{syn} with the number of words in all classes of W_{c_j} where $c_j \neq 1$. Then, the total mean overlap can be computed by:

$$1 - \frac{|W_{syn}|}{\sum_{c_j} |W_{c_j}|} = 0.292 \quad (4.2)$$

where $c_j \neq 1$.

Therefore, on average, there is a 29.2% overlap in words between the syntactic classes. We further break down the overlap calculations to show the overlaps between all pairs of classes in Table 4.6, including the semantic class W_{sem} in the first row.

From this table, we can compare the overlaps between the syntactic classes and the semantic class. We can see that there is a *disproportionately* large overlap between each syntactic class and W_{sem} . This is due to the fact that W_{sem} has a large number of words, and the odds that a word is in both the semantic class and a particular syntactic class are high. The total mean overlap between W_{sem} and a syntactic class W_{c_j} is calculated by:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
11111	43	696	8	397	51	133	49	9	141	11	71	78	41	621	36	131	212	177	543
	138	9	4	55	31	6	11	1	2	10	15	12	15	15	4	17	73	28	2
		1612	1	114	5	28	15	2	25	2	41	23	11	138	11	101	93	89	240
			19	7	14	1	1	0	2	2	1	0	3	9	0	11	7	1	4
				47	31	18	1	29	11	68	35	42	267	31	55	193	75	62	
					179	6	7	0	3	3	6	11	17	42	6	30	67	11	6
						287	5	0	11	0	13	4	15	37	2	8	27	8	25
							191	0	8	0	4	3	9	14	3	3	19	5	11
								43	0	0	1	0	0	3	0	2	1	0	2
									274	0	2	2	4	67	1	0	6	7	40
										29	2	1	1	1	1	2	6	4	2
											308	12	10	6	10	12	61	18	8
												280	5	14	7	9	28	15	19
													129	19	16	5	43	12	5
														1705	14	54	74	44	153
															144	8	13	9	5
																316	48	21	65
																	605	42	49
																		419	49
																			938

Table 4.6: Word overlaps between classes at $\eta = 0.9$. The bottom value for each column is the number of words in that particular class. The first column indicates W_{sem} and the first row indicates the overlap between each syntactic class and W_{sem} .

$$\frac{|W_{c_j} \cap W_{c_1}|}{|W_{c_j}|} = 0.361 \quad (4.3)$$

where $c_j \neq 1$.

Taking the average between W_{sem} and all the syntactic classes, we get the mean total overlap of 36.1%. Indeed, after merging all the syntactic classes into W_{syn} , we get a 36.7% overlap with W_{sem} , as well. This means that over 1/3 of the words in W_{syn} are also in W_{sem} . Essentially, there is a substantial set of semantic words in W_{syn} and as a result, in our final feature set.

Clearly, we need to improve the classification accuracy by reducing the influence of the semantic words in W_{syn} . In the next section, we have outline two *overlap reduction methods* to iteratively reduce the overlaps.

4.6 Semantic-Syntactic Overlap Reduction Methods

4.6.1 Set Difference

The first systematic approach (defined in 3.3.1) is derived by varying the percentage of top semantic words that should be excluded in the final feature set. The δ value is defined by the percentage of the words in W_{sem} that we exclude from W_{syn} . Note that we are using the same training parameters as we have done for the baseline approaches and the FS scheme based on syntactic classes. However, since we have been getting consistently better results using ‘eval’ number of iterations, we will forgo 10 and 25 iterations in favour of ‘eval’. The results for $0.0 \leq \delta \leq 1.0$ for increments of $\delta \times |W_{sem}|$ are summarized in Table 4.7.

From the results, we can see that the highest accuracy attained (at $\delta = 0.2$) is

δ	Mean	Std. Dev.	Overlap
0.0	0.861	7.7×10^{-3}	0.358
0.1	0.862	6.6×10^{-3}	0.338
0.2	0.865	5.8×10^{-3}	0.318
0.3	0.858	6.6×10^{-4}	0.296
0.4	0.857	1.1×10^{-2}	0.270
0.5	0.852	4.2×10^{-3}	0.242
0.6	0.846	2.0×10^{-3}	0.209
0.7	0.849	5.2×10^{-3}	0.168
0.8	0.847	3.7×10^{-3}	0.122
0.9	0.840	4.9×10^{-3}	0.065
1.0	0.831	3.7×10^{-3}	0.0

Table 4.7: Results for the Syntactic-Semantic class set difference method. Each row represents the accuracy achieved at the particular ϵ value.

only *marginally* higher than the results reported in Table 4.4. Employing a **one-tailed t-test** between the accuracy at $\delta = 0.2$ and our default HMM-LDA FS scheme at the 95% significance level, the resultant p-value of 0.6945 suggests no significant difference. Moreover, as we remove more and more words from W_{sem} , the accuracy level decreases. This suggests that $W_{sem} \cap W_{syn}$ contains some important features and if we subtract W_{sem} entirely, we essentially eliminate them. At each cutoff level, we are eliminating 10% until we have eliminated the whole set. Also, it is hard to pinpoint the exact δ value where the highest accuracy is attained. Finally, the overlap ratio is as expected, decreasing consistently as words in W_{sem} are removed. Clearly, a more fine-grained approach is needed, and that leads us to the Max-Score method.

4.6.2 Max Score Selections

For the method based on Max Scores, we may select features that are in both W_{sem} and W_{syn} sets as long as their max scores in W_{syn} are higher than those in W_{sem} . As a result, it is possible that high frequency features in W_{sem} are also included in the final feature set (see subsection 3.3.2 for the details of this scheme).

Iterations	Mean	Std. Dev.	Overlap
eval	0.875	6.6×10^{-3}	0.246

Table 4.8: Results for the Max Score Based FS method.

Comparing the accuracy in Table 4.8 with the accuracies in Tables 4.7 and 4.4, we can see that using the fine-grained Max-Score algorithm improves the classification performance. This means that iteratively removing words that have a relatively higher probability in W_{sem} compared to W_{syn} does not eliminate important words occurring in both sets, but lessens the influence of some high probability words in W_{sem} .

Once again, employing a **one-tailed t-test** between the accuracy achieved by the Max Score algorithm and our default HMM-LDA FS scheme, the resultant p-value of 0.0772 suggests that there is not a significant improvement in accuracy over our default FS scheme at the 95% significance level, but there is significant improvement at the 90% significance level. Furthermore, comparing this result to the baseline methods in Table 4.2, we obtain p-values of 0.011 and 0.014, which *do* indicate a significant improvement at the 95% significance level.

4.7 POS-assisted Semantic-Syntactic Overlap Reduction

Methods

With HMM-LDA, we can separate syntactic classes from the semantic class. However, not all syntactic classes are useful for SA, such as those corresponding to common stopwords and some nouns. In our final scheme (defined in 3.4), we consider our feature selection methods by restricting to a subset of the syntactic classes. We use POS tags as the differentiating factor to help us choose the classes we want to include in the subset. We first tag all the documents with POS tags and include only the classes that have

a ‘high’ percentage of overlaps with the selected POS categories such as adjectives and adverbs.

For our experiments, the selected tags are, $POS' = \text{'JJ', 'JJR', 'JJS', 'RB', 'RBR', 'RBS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'NN', 'NNS', 'NNP', 'NNPS'}$, which are mostly adjectives, adverbs, verbs, and nouns.

4.7.1 Overlaps with POS Tags

To establish a usable percentage of the tags in POS' , we look at the words in each class that have been tagged with the tags in POS' across all classes and tag groups. More specifically, for all classes $c_j \in C$ such that $1 \leq j \leq |C|$, we get the counts for each tag group in POS' :

$$\sum_{i=1}^N I(p_i \in POS') \times I(c_i = j) \quad (4.4)$$

where N is the total number of words, and p_i and c_i are the POS tag and syntactic class assigned for word w_i .

From Tables 4.9 and 4.10, we note that there are a significant number of nouns and a comparatively small number of adverbs. In fact, the average percentage of nouns per class is 54% and that for adverbs is 0.1%. Also, the classes that have a high number of target POS' groups such as JJ* and RB* generally also have a high number of total words. This is in line with our earlier understanding that HMM-LDA does not necessarily model syntactic structure based on POS as discussed in section 3.4.

In the next two sections, we repeat the schemes from sections 3.3.1 and 3.3.2, but instead of using all the syntactic classes for W_{syn} , we use only the classes that are returned from Algorithm 6.

Class	JJ*	RB*	VB*	NN*	Total words
0	2735	82	12936	144976	175205
1	978	45	4447	34360	46093
2	2369	70	9458	63200	82306
3	3781	114	14641	85903	120166
4	2304	319	11158	55565	80405
5	521	68	4227	14841	47917
6	80	0	1406	5091	20413
7	276	173	1886	20811	57187
8	0	0	135	14	30356
9	44	0	194	1944	19756
10	1548	34	5307	36894	66335
11	1387	15	10941	44844	89721
12	1651	21	2152	17885	102743
13	1820	552	5668	56803	105579
14	2714	287	9643	74369	95368
15	49	1747	3617	32708	41037
16	318	317	3078	19865	26112
17	2415	185	11512	60858	81819
18	248	5	1192	8864	22186
19	2163	162	6602	69416	83290

Table 4.9: POS tag percentage per class for POS groups (JJ*, RB*, VB*, NN*)

	JJ*	RB*	VB*	NN*
Mean per class	2.0×10^{-2}	1.0×10^{-3}	8.0×10^{-2}	5.4×10^{-1}
Std. Dev. per class	1.1×10^{-4}	9.0×10^{-5}	1.7×10^{-3}	7.1×10^{-2}
Num words in POS group	29745	4555	130570	920538

Table 4.10: First row: Mean POS tags per class. Second row: Tags per class Std. Dev. Third row: Total number of words per POS group.

4.7.2 FS Based on a Subset of Syntactic Classes

For our experiments, the restriction factor we have chosen for each class is the *mean per class*, calculated in 4.10. For example, if a class has a ratio of adjectives smaller than the mean, we do not include it. We can also vary the types of POS tags used. Therefore, the *POS'* set includes: 'JJ*', 'RB*' and 'VB*'. In Table 4.11, we see the subsets of classes used for each POS group, as well as the number of words after merging the subsets at $\eta = 0.9$.

POS Group	Classes	Number of words
JJ*	1, 2, 3, 4, 10, 11, 12, 13, 14, 17, 19	5317
JJ* + RB*	1, 2, 3, 4, 10, 11, 12, 13, 14, 15, 16, 17, 19	5490
JJ* + RB* + VB*	1, 2, 3, 4, 5, 10, 11, 12, 13, 14, 15, 16, 17, 19	5549

Table 4.11: Classes included after low percentage *POS'* classes removed.

δ	JJ*		JJ* + RB*		JJ* + RB* + VB*	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
0.0	0.864	6.4×10^{-3}	0.870	7.4×10^{-3}	0.865	7.7×10^{-3}
0.1	0.859	4.3×10^{-3}	0.863	2.3×10^{-3}	0.861	9.0×10^{-3}
0.2	0.863	5.6×10^{-3}	0.860	2.8×10^{-3}	0.860	8.0×10^{-3}
0.3	0.857	4.3×10^{-3}	0.857	8.3×10^{-3}	0.854	3.6×10^{-3}
0.4	0.855	5.4×10^{-3}	0.849	4.1×10^{-3}	0.847	1.2×10^{-3}
0.5	0.855	3.0×10^{-3}	0.854	8.8×10^{-3}	0.839	3.4×10^{-3}
0.6	0.852	4.8×10^{-3}	0.844	7.3×10^{-3}	0.845	5.6×10^{-3}
0.7	0.843	5.7×10^{-3}	0.846	6.5×10^{-3}	0.839	8.0×10^{-3}
0.8	0.847	8.3×10^{-3}	0.838	9.7×10^{-3}	0.844	8.1×10^{-3}
0.9	0.847	2.7×10^{-3}	0.842	5.8×10^{-3}	0.832	4.4×10^{-3}
1.0	0.837	1.8×10^{-3}	0.839	9.9×10^{-3}	0.822	1.5×10^{-3}

Table 4.12: Results for the POS-aided method by POS categories in the included set *POS'*.

Iterations	JJ*		JJ* + RB*		JJ* + RB* + VB*	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
eval	0.869	5.9×10^{-3}	0.873	1.2×10^{-3}	0.872	1.8×10^{-3}

Table 4.13: Result for the POS-aided Max-Score method by POS categories in the included set *POS'*.

From Tables 4.12 and 4.13, we do not see a significant improvement in accuracy by removing certain syntactic classes. To justify this claim, we compute the p-value for the highest accuracy level between the default Set Difference scheme and the POS-assisted Set Difference (-NN*) scheme. Similarly, we compute the p-value for the mean accuracy between Max Score and POS-assisted Max Score (JJ* + RB*). At the 95% significance level, the p-values of 0.4091 and 0.7155 show that there is no significant difference.

First of all, for the JJ* group, there are 11 classes that are used and 8 classes that have been held out. In particular classes 5, 6, 7, 8, 9 and 15 have a very low number

of adjectives and if look at the table of overlaps (see 4.9), we can see that the other classes have contributed the adjectives that are missing if we take out these classes. Secondly, there is not a significant number of adverbs to make a difference in the accuracy levels. Lastly, if we take into account all the verbs, there are now 15 classes used out of a total of 19. Since classes 6, 7, 8 and 9 all have a comparatively low ratio of tags from POS' , this does not make a difference in the accuracy. Also, the number of classes and the accuracy levels are bound to the original 19 classes. Therefore, including more classes will ensure that the accuracy level approaches that for all 19 classes.

4.8 Discussion

From the experiments above, the best accuracy is achieved by utilizing the Max-Score algorithm (see 3.3.2 and 4.6.2) with a 2500 df cutoff from W_{syn} . As we have mentioned in 3.3.2, the Max-Score algorithm enables us to lessen the influence of the words from W_{sem} by iteratively removing words that have a high probability in W_{sem} . The overall result is that we can improve our feature set by taking out words that have a high probability in W_{sem} while allowing the other words that could be in W_{sem} that have a high probability in one of the W_{syn} classes.

To quantify our arguments, we have calculated a series of t-tests after each experimental result. Our calculated p-values for one-tailed t-tests when comparing our best approach with the DF and POS (-NN*) baselines suggest that the classification accuracy of our best approach is *significantly* higher than the baseline approaches at the 95% level of significance.

Utilizing our Maximum Entropy classifier, we can gauge what features are the most useful in identifying document polarity after training. If we train the classifier on

the entire corpus (all 2000 documents) using the best feature set in section 4.6.2, the top 25 best features are in Table 4.14:

Feature	Polarity	Odds
outstanding	pos : neg	11.5 : 1.0
ludicrous	neg : pos	11.0 : 1.0
finest	pos : neg	8.1 : 1.0
idiotic	neg : pos	7.4 : 1.0
wonderfully	pos : neg	7.0 : 1.0
breathtaking	pos : neg	6.6 : 1.0
anger	pos : neg	6.5 : 1.0
inept	neg : pos	5.5 : 1.0
wasted	neg : pos	5.5 : 1.0
lame	neg : pos	5.5 : 1.0
beautifully	pos : neg	5.4 : 1.0
awful	neg : pos	5.3 : 1.0
poorly	neg : pos	5.1 : 1.0
ridiculous	neg : pos	5.0 : 1.0
garbage	neg : pos	4.8 : 1.0
uninspired	neg : pos	4.7 : 1.0
era	pos : neg	4.7 : 1.0
lifeless	neg : pos	4.6 : 1.0
random	neg : pos	4.5 : 1.0
worst	neg : pos	4.4 : 1.0
refreshing	pos : neg	4.3 : 1.0
mature	pos : neg	4.3 : 1.0
whatsoever	neg : pos	4.3 : 1.0
bland	neg : pos	4.2 : 1.0
painfully	neg : pos	4.2 : 1.0

Table 4.14: Top 25 features given Max Score algorithm (3.3.2) with all syntactic classes. The last column shows the polarity odds per feature.

From Table 4.14, it is clear that adjectives and adverbs are the best indicators of polarity because given a feature set with a mixture of POS groups, the *presence* of particular adjectives and adverbs that convey a certain polarity are the most useful features for a Maximum Entropy classifier. However, we also see some nouns such as ‘garbage’ (used metaphorically to associate negativity) and ‘random’ (used to indicate confusion and disarray) appear as well. These features fall in line with the feature characteristics that were discussed in 3.1.2.

Chapter 5

Conclusions and Future Directions

5.1 Conclusions

We approached the task of Feature Selection for Sentiment Analysis by using a Content and Syntax model, known as HMM-LDA, to separate the *entities* in a review document from the potentially sentiment carrying *modifiers*. HMM-LDA models entities and modifiers as long-range and short-range dependencies, allowing us to separate words into semantic and syntactic classes, respectively. By grouping all the sentiment modifiers for the entities in a document into the syntactic classes, we are selecting the features that are intuitively in line with the outlined characteristics of salient features for SA (see subsection 3.1.2).

In subsection 3.2, we introduced the method of training an HMM-LDA model with a Gibbs sampler and combining the resultant syntactic classes into a set of potentially sentiment-carrying words. The results obtained from these experiments (see subsection 4.4) showed an improvement in accuracy versus the baseline approaches. This suggests that by using only the syntactic classes (which correspond to the modifiers), we can immediately see an improvement in the classification accuracy for documents.

The two algorithms (Algorithm 4 and Algorithm 5 in subsections 3.3.1 and 3.3.2) were introduced as a way of reducing the influence of the semantic words in our final feature sets. We deemed this necessary because semantic words do not usually carry

sentiment and mostly correspond to *neutral* entities. In the classification experiments done in subsection 4.6.1 for Algorithm 4, we can see that the accuracy level increases slightly before eventually decreasing. As this scheme iteratively removes a percentage of features appearing in the semantic class, it reaches a point where the highest accuracy is attained before it starts to remove too many salient features, which is reflected in the decrease in accuracy. As discussed, this approach has two deficiencies: The highest accuracy level obtained is only marginally higher than the simple approach of only using syntactic classes, and it is hard to pinpoint the exact cutoff level where the accuracy is the highest.

This motivated us to implement a more fine-grained approach that enables us to compare one feature at a time. The results obtained in subsection 4.6.2 show a greater improvement in the classification accuracy over the baseline approaches, and also the approach based solely on syntactic classes. Essentially, this improvement in accuracy can be attributed to the removal of some of the non-sentiment-carrying features from the semantic class and replacing them with more appropriate sentiment-carrying features from the syntactic classes.

In our final FS scheme, we placed higher importance on some syntactic classes over others. Since sentiment-carrying words generally consist of adjectives and adverbs, we only include the classes in our final feature set that have a high overlap with these parts-of-speech groups while ignoring the rest. We then generated the results based on the two feature selection algorithms (see subsections 4.6.1 and 4.6.2). The reason we did not see a significant improvement in classification accuracy suggests that the HMM-LDA model does not assign words to classes based on the POS tags, but rather on the observed syntactic relationships in the documents.

Overall, the increase in accuracy over the baseline approaches in all of our FS

schemes can be explained by the discriminative nature of the modifying words that our schemes pick up. One of the main characteristics for salient features is that they must be discriminative enough so that the learning system is able to pick up on their presence in certain documents relating to one outcome and the absence in other documents relating to another outcome. Since the HMM-LDA syntactic classes by and large pick up modifying words that describe entities, these words by definition cannot be both positive and negative at the same time, given the domain. It is then important to minimize the impact of *neutral* (non-sentiment-carrying) features. In all our approaches, we minimize the impact by separating the semantic class from the syntactic classes, and as a result, removing some of the neutral features that are present in the baseline schemes.

5.2 Future Directions

We have thus far only experimented on our feature selection schemes with the movie review domain, since the data are readily available, pre-processed, and annotated. Moreover, according to Pang et al. [29], the accuracy levels achieved for movie reviews are generally lower than those for product reviews and hotel reviews. This is due to the homogeneous nature of movie reviews where reviewers evaluate the characteristics of the entities in movies using more expressive language. It will be interesting to test our approach for other domains. For example, to evaluate our FS schemes on product reviews, we would have to train an HMM-LDA model on that specific domain and use a maximum entropy classifier to evaluate the sentiment classification accuracy as we have done for the movie reviews.

One thing to note is that HMM-LDA is not necessarily restricted to unigrams. In fact, the vocabulary can consist of unigrams, bigrams, trigrams, and so on. To train an

HMM-LDA model on *n-grams*, we would simply alter our parsing algorithm to extend the vocabulary to occurrences of multiple words. Useful n-grams typically consist of adverb-adjective pairs such as ‘too small’ and ‘very unhappy’ and negation pairs such as ‘not good’, which are clear indicators of sentiments.

Another avenue for future development of this framework could include identifying and extracting *aspects* from a review document. So far, we have not identified aspects from the entities, choosing instead to classify a document as a whole. However, this framework can be applied to extract relevant (most probable) aspects using the LDA topic model and then restrict the syntactic modifiers to the range of sentences where an aspect occurs. In other words, we can extend our model to make use of the LDA topics, which would map directly to the aspects in a review. Since the aspects are mixed throughout the document, for each aspect we would restrict the modifiers to only the sentences in which the aspect occurs. This would give us a tuple consisting of the words making up the aspect (from the LDA topic model) and the modifiers that relate the aspect to a polar category. After that, we can use these modifiers to train a maximum entropy model and then evaluate the classification accuracy. This would give us an *unsupervised* aspect extraction scheme that we can combine with a classifier to predict polarities for each aspect.

Finally, we can extend our framework to classify review documents based on a *scale*. Thus far we have focused on the *binary* classification task. A review document can be either ‘positive’ or ‘negative’. However, many applications need a further level of detail such as ‘how positive’ and ‘how negative’. By adding a further level of categorization in the form of a scale (for example 1 to 5, where 1 is ‘very bad’ to 5 being ‘very good’), we can use the same features that we have gathered from the HMM-LDA model but simply add more categories for the maximum entropy classifier. Of course, we would need the

necessary annotated data to accomplish this task and the accuracy levels may be reduced simply because the classifier would have more categories to handle.

Bibliography

- [1] Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with python: Analyzing text with the natural language toolkit. pages 221–257, Beijing, 2009. O’Reilly.
- [2] C. M. Bishop. Pattern recognition and machine learning. pages 423–450, New York, 2006. Springer.
- [3] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era (NLPiX)*, pages 514–522, 2008.
- [4] David M. Blei and Jon D. McAuliffe. Supervised topic models. In *NIPS*, pages 893–901, 2007.
- [5] David M. Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.
- [6] Jihong Cai and Fei Song. Maximum entropy modeling with feature selection for text categorization. In *AIRS*, pages 549–554, 2008.
- [7] Jonathan Chang, Jordan Boyd-graber, Sean Gerrish, Chong Wang, and David M. Blei. Reading tea leaves: How humans interpret topic models, 2009.
- [8] Paula Chesley, Bruce Vincent, Li Xu, and Rohini Srihari. Using verbs and adjectives to automatically classify blog sentiment. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, pages 27–29, 2006.
- [9] Sanjiv R. Das and Mike Y. Chen. Yahoo! for Amazon: Sentiment extraction from small talk on the Web. *Management Science*, 53(9):1375–1388, 2007.
- [10] Xiaowen Ding, Bing Liu, and Philip S. Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the Conference on Web Search and Web Data Mining (WSDM)*, pages 231–240, 2008.
- [11] A. Gelman, J. Carlin, H. Stern, and D. Rubin. Bayesian data analysis, second edition. page 582. Chapman & Hall/CRC, 2003.
- [12] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *PNAS*, 101(suppl. 1):5228–5235, 2004.
- [13] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, 2005.
- [14] Vasileios Hatzivassiloglou and Kathleen McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the Joint ACL/EACL Conference*, pages 174–181, 1997.

- [15] Vasileios Hatzivassiloglou and Janyce Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 299–305, 2000.
- [16] Gregor Heinrich. Parameter estimation for text analysis. Technical report, 2004.
- [17] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 289–29, San Francisco, CA, 1999. Morgan Kaufmann.
- [18] John A. Horrigan. Online shopping. Pew Internet & American Life Project Report, 2008.
- [19] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *Proceedings of AAAI*, pages 755–760, 2004.
- [20] Matthew Hurst and Kamal Nigam. Retrieving topical sentiments from online document collections. In *Document Recognition and Retrieval XI*, pages 27–34, 2004.
- [21] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1367–1373, 2004.
- [22] David D. Lewis, Fan Li, Tony Rose, and Yiming Yang. Reuters Corpus Volume 1 as a text categorization test collection. *Journal of Machine Learning Research*, 5:361–397, April 2004.
- [23] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceeding of the 18th ACM conference on Information and knowledge management, CIKM '09*, pages 375–384, New York, NY, USA, 2009. ACM.
- [24] Christopher D. Manning and Hinrich Schütze. Foundations of statistical natural language processing. pages 589–596, Cambridge, MA, USA, 1999. MIT Press.
- [25] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 412–418, July 2004. Poster paper.
- [26] Kamal Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.
- [27] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 271–278, 2004.
- [28] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124, 2005.
- [29] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2:1–135, January 2008.

- [30] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.
- [31] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 25–32, 2003.
- [32] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
- [33] Mark Steyvers and Tom Griffiths. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning.*, pages 427–446. Laurence Erlbaum, 2006.
- [34] I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *International World Wide Web Conference (WWW)*, pages 111–120, 2008.
- [35] Peter Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 417–424, 2002.
- [36] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346, 2003.
- [37] Yi Wang. Distributed gibbs sampling of latent topic models: The gritty details. Technical report, 2008.
- [38] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM)*, pages 625–631. ACM, 2005.
- [39] Janyce Wiebe. Learning subjective adjectives from corpora. In *Proceedings of AAAI*, pages 277–308, 2000.
- [40] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 347–354, 2005.
- [41] Jun Yan, Ning Liu, Benyu Zhang, Shuicheng Yan, Zheng Chen, Qiansheng Cheng, Weiguo Fan, and Wei ying Ma. Ocfs: Optimal orthogonal centroid feature selection for text categorization. In *In Proc of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 122–129, 2005.
- [42] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97*, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

- [43] Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 427–428, 2003.