Representation and Learning Schemes for Sentiment Analysis

Rahman Mukras

A thesis submitted in partial fulfilment
of the requirements of
The Robert Gordon University
for the degree of Doctor of Philosophy

January, 2009

Bismillah

Dedicated to my Parents, Zamzam, Hafswa,

Mariam, Suleiman, and Saad.

I Love you all Dearly

Abstract

This thesis identifies four novel techniques of improving the performance of sentiment analysis of text systems. Thes include feature extraction and selection, enrichment of the document representation and exploitation of the ordinal structure of rating classes. The techniques were evaluated on four sentiment-rich corpora, using two well-known classifiers: Support Vector Machines and Naïve Bayes.

This thesis proposes the Part-of-Speech Pattern Selector (PPS), which is a novel technique for automatically selecting Part-of-Speech (PoS) patterns. The PPS selects its patterns from a background dataset by use of a number of measures including Document Frequency, Information Gain, and the Chi-Squared Score. Extensive empirical results show that these patterns perform just as well as the manually selected ones. This has important implications in terms of both the cost and the time spent in manual pattern construction.

The position of a phrase within a document is shown to have an influence on its sentiment orientation, and that document classification performance can be improved by weighting phrases in this regard. It is, however, also shown to be necessary to sample the distribution of sentiment rich phrases within documents of a given domain prior to adopting a phrase weighting criteria.

A key factor in choosing a classifier for an Ordinal Sentiment Classification (OSC) problem is its ability to address ordinal inter-class similarities. Two types of classifiers are investigated: Those that can inherently solve multi-class problems, and those that decompose a multi-class problem into a sequence of binary problems. Empirical results showed the former to be more effective with regard to both mean squared error and classification time performances.

Important features in an OSC problem are shown to distribute themselves across similar classes. Most feature selection techniques are ignorant of inter-class similarities and hence easily overlook such features. The Ordinal Smoothing Procedure (OSP), which augments inter-class similarities into the feature selection process, is introduced in this thesis. Empirical results show the OSP to have a positive effect on mean squared error performance.

Acknowledgements

I firstly wish to thank my supervisors Dr. Nirmalie Wiratunga and Dr. Robert Lothian. You always went an extra mile to offer me support when I most critically needed it. I am deeply indebted to you for this.

I am grateful to Dr. Sutanu Chakraborti who played an integral part in helping to finally break the ice with a number of the methodologies presented in this thesis; to Amandine Orecchioni and Dr. Stewart Massie for giving me the privilege to contribute to their research; to Dr. Ivan Koychev and Prof. David Harper who used to form part of my supervisory team; to Muhammed, Siddhartha, Ganesan, Zia, Kefang, Ibrahim, Ibrahim Alwawi, Stella, Ratiba, Daniel, Peng, Jean-Claude, David, Thierry, Peter, Sunny, Guofo, Bayo, Selphi, Bhavani, Ralph, Sandy, Malcolm and Ulises who helped in making the CTC a vibrant research community; to Prof. Susan Craw and all members of her KBS group for providing vital feedback for this research; to Prof. John Carroll, my MRes supervisor at Sussex University, for introducing me to the world of Text; and finally to all those who played a part in making this research possible, Thank you.

Contents

1 Introduction **1**

1.1 The Challenges of Sentiment Analysis 2

1.2 Sentiment Analysis of Text 2

1.2.1 Natural Language Processing 3

1.2.2 Information Retrieval 3

1.2.3 Text Classification 4

1.3 Objectives of this Research 5

1.4 Overview of the Thesis 6

2 Literature Survey **8**

2.1 Identifying the Polarity of a Word 8

2.1.1 Un-Supervised Approaches 9

2.1.2 Supervised Approaches 12

2.2 Identifying the Polarity of a Document 15

2.2.1 Un-supervised Approaches 16

2.2.2 Supervised Approaches 17

2.3 Summary 18

3 Background **20**

3.1 The Datasets 21

3.1.1 The Problem Datasets 21

3.1.2 The Support Datasets 23

3.2 Textual Pre-Processing 24

3.3 Feature Selection 25

3.4 The Classifiers 28
3.4.1 Naïve Bayes 29
3.4.2 The Support Vector Machine 31
3.5 Performance Metrics 32
3.5.1 Mean Squared Error 33
3.5.2 Comparing Performance 33
3.6 Summary 34

4 Sentiment Extraction with PoS Pattern Analysis 35
4.1 Overview of the System 38
4.1.1 Pattern Selection from a Background Dataset 39
4.1.2 Extraction of Phrases 39
4.1.3 Un-supervised Sentiment Classification 40
4.2 Discussion and Summary 43

5 Positional Information 44
5.1 The Effects of Positional Information 45
5.2 Weighting Phrases by Position 47
5.3 Discussion and Summary 48

6 A Comparative Study of OSC 50
6.1 Multi-Class Classifiers for OSC Problems 52
6.1.1 The One-vs-All Scheme 52
6.1.2 The All-vs-All Scheme 53
6.1.3 The Simple Ordinal Approach 54
6.1.4 The Binary Search Approach 55
6.1.5 The Single Classifiers 57
6.2 Discussion and Summary 58

7 Feature Selection for OSC Problems 60
7.1 Inter-Class Similarities 62
7.1.1 A Solution by Modifying Information Content 62
7.1.2 The Ordinal Smoothing Procedure 63

7.2 Discussion and Summary 66

8 Evaluation **67**

8.1 Evaluation of the PPS Algorithm 68

8.1.1 Experiment 1: Comparison of Pattern Selection 69

8.1.2 Experiment 2: Manual vs. Automated Pattern Selection 70

8.1.3 Experiment 3: Effects of a Sentiment Rich Background 71

8.2 Evaluation of Positional Information 73

8.3 A Comparative Study on OSC 74

8.3.1 Experiment 1: Comparison of MSE Performance 74

8.3.2 Experiment 2: Comparison of Classification Speed 80

8.4 Evaluation of the OSP 81

8.5 Summary 84

9 Conclusions and Future Work **85**

9.1 The Objectives Revisited 85

9.2 Future Work 88

9.2.1 Beyond Tag Adjacency 88

9.2.2 Beyond OSC 89

A General Appendix **95**

A.1 Comparing Performance 95

A.2 Supplementary Results 96

A.3 Opinion Retrieval of Blogs 97

A.3.1 The Opinion Retrieval System 100

A.3.2 Implementation 102

A.3.3 Discussion and Summary 103

A.4 Textual Pre-Processing 103

A.4.1 Part-of-Speech Tagging 104

A.4.2 Stemming 104

A.4.3 Conversion to a Consistent Letter Case 105

A.4.4 Stop-Word Filtering 106

A.5 List of Abbreviations 109
A.6 Published Papers 110

List of Figures

1.1 The Objectives within a Typical SAoT Research Framework 5

2.1 The Problem with the Frequency Hypothesis 14

3.1 The Textual Pre-Processing Pipeline 24
3.2 The Stages in Textual Pre-Processing 25
3.3 A Contingency Table for an m Class Problem 27
3.4 Support Vector Machines: Classification and Regression 31
3.5 Inter-Class Euclidean Distance for each class of the Actors Dataset 33

4.1 The PPS Algorithm in a Sentiment Classification Framework 37
4.2 Extraction of Phrases from the Test Document d 39
4.3 The Un-Supervised Sentiment Orientation Algorithm 42

5.1 Mean Distributions of Sentiment Rich Words across a Document 46
5.2 Performances of Four Main Sections of a Document 47

6.1 The One-vs-All Classifier 52
6.2 The All-vs-All Classifier 53
6.3 The Simple Ordinal Approach 54
6.4 The Binary Search Approach 56

7.1 Feature f_1 and f_2 posses Distributions that are Equally Important 61
7.2 Feature f_1 and f_2 are Distinguished by Ordinal Smoothing 62
7.3 The OSP as Applied to a given Feature f_t 63
7.4 Linear and Exponential Approximations of Inter-class Similarity 64

8.1 Comparison of the Pattern Selection Techniques on Actors 68
8.2 Comparison of the Pattern Selection Techniques on Edmunds 69
8.3 Manual vs. Automated Pattern Selection 70
8.4 Performance of the Weighting Functions on the Datasets 75
8.5 MSE of SVM-reg and Naïve Bayes based classifiers on OSC 76
8.6 Confusion Matrices for OvA on Actors and Edmunds 77
8.7 Comparison of SVM-reg and SOA on the Scale Dataset 78
8.8 Confusion Matrices for SOA on Edmunds 79
8.9 IG between the Extreme Positive and all Other Classes 83

A.1 The Impact of the OSP while using SVM-reg 97
A.2 The Impact of the OSP while using Naïve Bayes 98
A.3 Performance of the Weighting Functions on the Actors Dataset 99
A.4 Performance of the Weighting Functions on the Edmunds Dataset 100
A.5 An Overview of the Opinion Retrieval System. 101
A.6 The Textual Pre-Processing Pipeline 102
A.7 A Standard Stop Word List 107

List of Tables

2.1 Comparison of Supervised and Un-Supervised Techniques [Pang et al., 2002] . . 13

3.1 Information and Statistics about the Datasets 21

4.1 Manually Selected PoS Patterns 36
4.2 Sample of top ranked PoS patterns selected using IG CHI and DF 38
4.3 A Sample of the Adjectives used to Generate the Sets $\mathcal{P}$ and $\mathcal{N}$ 40

8.1 The Performance of Sentiment Rich Background Data on Actors (uni-tags) . . . 71
8.2 The Performance of Sentiment Rich Background Data on Edmunds (uni-tags) . . 72
8.3 The Performance of Sentiment Rich Background Data on Actors (mix-tags) . . . 73
8.4 The Performance of Sentiment Rich Background Data on Edmunds (mix-tags) . . 74
8.5 Time Performance in Milliseconds Per Cross-Validation Fold 80
8.6 The Impact of the OSP while using SVM-reg 81
8.7 The Impact of the OSP while using Naïve Bayes 82

A.1 The Effect of Part-of-Speech Tagging 103
A.2 The Effect of Stemming 105
A.3 The Effect of using a Consistent Letter Case 106
A.4 The Effect of a Modified Stop-Word List 108

Chapter 1

Introduction

The simplicity of Internet publishing has resulted in users postings up their thoughts and sentiments in a variety of different forms with incredible prolificacy. Furthermore, much of these postings remain largely unmonitored. Blogs (short for web-logs), for instance, are particularly rich in sentiment and are now published by millions of web-users on a daily basis. This has led them to be considered as the latest form of self expression, and it is evident from their ever increasing mass that they are changing the face of the web. Message boards and Newsgroups have also been used to post comments on various issues and it is possible to track specific threads of discussion for over several months. There also exists a number of opinion sites such as Epinions.com, PlanetFeedback.com, and Rateitall.com which focus solely in collecting both professional and amateur reviews on numerous issues ranging from basic home-ware to corporate level systems.

Aside from adding to the mass of sentiments already present on the web, there have also been a few attempts to mine these opinion rich resources as well. For instance, companies such as BuzzMetrics.com offer the service of tracking web-users who initiate trends among thousands of consumers online. This clearly has a great potential in the world of advertising. There also exists specialised search engines such as BlogPulse.com, Technorati.com, and BlogLines.com which monitor and index blog posts on a daily basis. This enables users to search and tune into the latest *buzz*[1] within the blogosphere[2]. Furthermore, aside from the basic search, there also exists extra analysis tools such as the "Trend Search" by BlogPulse.com which allows users to create graphs that visually track the online buzz over time. One can, for instance, compare the online buzz

[1] An interesting discussion shared by many.

[2] A term that encompasses all blogs on the Internet.

between “Hillary” and “Obama.” A hoard of other similar opinion mining functionalities can also be found on the web.

1.1 The Challenges of Sentiment Analysis

It is interesting to note that, although many of the previously discussed systems seem adequate for opinion based search, a closer look would reveal that the type of search they conduct is geared towards mining *topic* rather than *sentiment*. For instance, when queried with the search term “Tea,” three of the top ten posts returned by a popular opinion search engine were entitled: “The History of Tea,” “A History of the Nations Favourite Beverage,” and “How White Tea Is Graded.” Notice that these posts are likely to have tea as their topic rather than as their main sentiment focus. Ideally, the list should have contained titles such as: “I Just Love Lebanese Tea,” or “I Cannot Stand Milk Tea.” Another illustration of where these systems deviate is in their type of results. Most of the time, they return only one ranked list relating to the most relevant documents to the query. This is not ideal for a sentiment oriented search, because sentiment is not based on relevance but rather on polarity, i.e. how positive or negative the documents are with respect to the query. Consequently, a better option would be to return either two ranked lists, positive and negative, or offer the option of reversing a single bi-polar list.

The problems mentioned above are most likely due to the direct importation of standard search technology into solving Sentiment Analysis of Text (SAoT) problems. They highlight at least two of the many issues that differentiate SAoT from other closely related disciplines, i.e. the difference between a “Sentiment and a Topic,” and between “Polarity and Orthogonality.”

In order to understand how to best address the challenges raised by SAoT, it is necessary to first take a closer look at what it actually entails. It then becomes possible to borrow ideas from other related disciplines or develop new SAoT solutions from scratch.

1.2 Sentiment Analysis of Text

SAoT is a research area that generally aims to determine the opinion expressed within a given text. Amongst some of the relevant areas that it covers are Natural Language Processing (NLP), Information Retrieval (IR) and Text Classification. Each of these disciplines have an important

significance to SAoT and the following sections provide a brief overview of how they integrate into the field.

1.2.1 Natural Language Processing

A sentiment has the interesting characteristic of being closely associated with certain Part-of-Speech (PoS) patterns. For instance, a PoS pattern containing an adjective followed by a noun is almost always an indicator of a sentiment rich phrase, e.g. "dishonest man," "rude child," or "generous host." It consequently follows that SAoT would be closely related to some of the techniques developed in NLP such as PoS tagging which is the process of marking up words within a given text as corresponding to a particular PoS. A standard PoS tagset can consist of in excess of a hundred PoS tags, which in turn means that several different combinations of PoS patterns are possible. The work done by Turney [2002] explores the use of PoS patterns in extracting sentiment rich phrases. He, however, only uses a set of five manually designed patterns, which is a small number compared to the various combinations that are possible. Nonetheless, his patterns were carefully crafted, and previous studies have found them to fair well against automatically generated patterns of greater number [see Mukras et al., 2007a].

1.2.2 Information Retrieval

IR Technology [Rijsbergen, 1979] has also had a significant impact on SAoT research. It is centred around the problem of identifying a set of documents, from amongst a larger collection, which are most relevant to a given query. IR has had its greatest impact on the web in the form of search engines such as Google, Yahoo, or AltaVista. The importance of IR to SAoT research is mainly due to the fact that online information defines the main source of sentiment rich text. This is evident from web applications such as Blogs, Newsgroups, Bulletin Boards, Web-diaries and Review portals which all contain an abundance of sentiment rich information. In addition to this, the opinions expressed in such web applications are often "in sync" with current affairs, and this factor makes them quite useful to industries such as politics, marketing, or the media where direct user feedback is an invaluable resource. Such are the factors behind the motivation that drives the development of IR systems for SAoT problems.

However, despite the advancements in IR Technology, the vast majority of sentiment rich text

on the web still remains hardly searchable. There are a number of factors that can be attributed to this, most of which revolve around the complex issue of "relevance." As was specified earlier, most IR systems are designed with the main objective of finding documents that are deemed to be "relevant" to a users query. However, the precise metric of relevance is open to discussion. Up to recent times more emphasis was given to the presence or absence of the query terms in a document rather than on the linguistic structures that the query terms resided in. The implication of this is that standard IR systems became ill-equipped to decode language constructs, such as sentiment, which require alternative methodologies in order to be manifested. Upon realising this, the IR research community decided to take a number of measures to bridge the IR-to-SAoT gap. Amongst the main ones was the initiation of the Blog Track into the Text Retrieval Conference (TREC) of 2006, which has now carried on to 2007 and 2008. The track has traditionally maintained the opinion retrieval task which involves the retrieval of documents that are opinionated about 50 predefined topics. The documents are retrieved from the TREC-Blog collection which consists of over 3,000,000 blog posts [see Macdonald and Ounis, 2006]. Progress in this area has been understandably minimal. Nonetheless, the competing systems seem to improve with successive conferences and the future could hold promising results.

1.2.3 Text Classification

Text Classification involves the task of automatically classifying a set of documents into a set of predefined classes. This is mostly done using Supervised Learning, which is a technique based on Machine Learning Technology [Mitchell, 1997]. In the context of SAoT, a supervised learning algorithm would be trained on a set of sentiment classified training documents. The documents are typically represented as vectors that lie within a space whose dimensions correspond to a sub-set of selected features[3] from the original training documents. Once training is complete, the algorithm would then be expected to correctly predict the class of a previously unseen test document that follows the same document-to-label distribution as the training set.

One of the major differences between standard SAoT and text classification is that the former normally consists of problems with ordinal classes, whereas those of the latter type are largely orthogonal. A problem is said to be ordinal if the similarity between any two of its classes decays with the linear distance between them. Such problems are referred to here as Ordinal Sentiment

[3] A feature refers to either one word or a group of words.

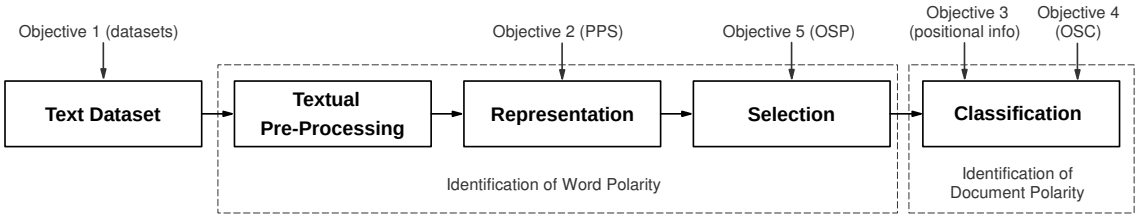

Figure 1.1: The Objectives within a Typical SAoT Research Framework

Classification (OSC) problems. As for an orthogonal problem, its classes bear equal inter-class similarities and thus have no explicit ordering. Most text classification problems are orthogonal, and hence most of the methodologies designed for them are typically ignorant of inter-class similarities. A number of studies have illustrated that significant gains can be yielded by taking inter-class similarities into account in both feature selection [Mukras et al., 2007b] and OSC [Frank and Hall, 2001]. This is therefore one of the avenues that this research aims to explore.

1.3 Objectives of this Research

Despite the fact that active SAoT research only commenced within the current decade, a considerable amount of ground has been covered. There now exists a variety of sub-problems in SAoT, as opposed to the mid 90's where most studies were based on predicting either the orientation of adjectives or the subjectivity of text [Hatzivassiloglou and McKeown, 1993 1997; Wiebe, 1994; Wiebe et al., 1999]. Nonetheless, much work still lies ahead as our understanding of how to solve these newly unearthed problems is still quite limited.

Figure 1.1 provides a generic overview of a typical SAoT research framework. It also highlights the areas that are addressed by the objectives of this research. The first step is to obtain a textual dataset with the correct properties (e.g. contains ordinal classes). This dataset is then pre-processed prior to mapping it onto a suitable representation. Feature selection is then performed, after which the text is finally classified. Steps 2, 3, and 4 generally aim at identifying the polarity of words (or features), whereas step 5 aims at identifying document polarity. Having discussed the general SAoT research framework, the following are the six objectives that will be addressed in this thesis:

1. *To compile two ordinal SAoT datasets from the real-world:* There are currently very few ordinal SAoT datasets reported in the literature (only one know by Pang and Lee [2005]).

It is thus important to provide supplementary datasets that will support the experimental findings of this research.

2. *To research on a new technique of automatically selecting PoS patterns to be used for extracting sentiment rich text from test documents:* The aim of this is to overcome the need for manual generation of PoS patterns.

3. *To investigate the possible approaches of weighting the importance of a phrase with respect its position within a document:* The hypothesis here is that different regions of a document contain different levels of sentiment. The goal is thus to exploit these differences by use of a weighting scheme.

4. *To perform a comparative study of multi-class classification techniques as applied to OSC problems. To also introduce and compare a new technique, purpose designed for OSC:* Most studies on OSC have only adopted a small subset of the possible multi-class classifiers. This research aims to perform a more extensive study of the various possibilities, which also includes a new technique that is purpose designed for OSC.

5. *To implement a novel strategy for selecting features from OSC problems:* OSC problems posses ordinal inter-class similarities and most feature selection techniques are not designed to accommodate this.

6. *To evaluate the methodologies mentioned in objectives 2-5:* This involves a thorough empirical evaluation.

A secondary objective of this research is to develop an Opinion Retrieval System that will compete in the Blog Track of the Text Retrieval Conference. Note however that the demands of this conference are more inclined towards competitive performance, rather than research excellence, and hence this objective is mentioned separately from the rest.

1.4 Overview of the Thesis

The next chapter presents a survey of SAoT in light of the objectives that will be addressed in this research. Chapter 3 then presents the methodological framework of the experiments that are to

be conducted in the research. It describes the datasets, textual pre-processing techniques, feature selection heuristics, classifiers, and performance measures that were employed.

Chapter 4 discusses a technique that automatically generates PoS patterns for extracting sentiment rich phrases from a test document. The objective here is to offer a less expensive alternative to manual pattern construction. Chapter 5 then looks at how the phrases that have been extracted by the patterns can be weighted with respect to their location within the document.

Chapter 6 discusses various multi-class classification techniques as applied to OSC problems. The chapter also introduces the Binary Search Approach (BSA) which is a technique that is purpose designed for OSC problems. The BSA recursively splits the training set of an OSC problem into equally sized halves, hence enabling it to maintain the original linear ordering at each recursion level. This aspect is immensely important when solving OSC problems.

Chapter 7 presents a new strategy for selecting features for OSC problems. The chapter first highlights the inability of standard feature selection techniques to utilise the ordinal information within OSC problems. It then goes on to discuss the Ordinal Smoothing Procedure (OSP) which tackles this problem explicating ordinal information, hence making it possible for standard feature selection techniques to efficiently handle OSC problems.

All the methodologies presented in Chapter 4, 5, 6, and 7 are then evaluated in Chapter 8. Note also that an account of the system that was developed for the Opinion Retrieval Task in the Blog Track of the 2007 TREC is presented in the General Appendix. The thesis is finally concluded in Chapter 9 with a discussion on contributions and future directions.

Chapter 2

Literature Survey

The aim of this chapter is to present a brief literature survey of the domain and its potential for future scientific research. In addition to this, it also serves the purpose of introducing some of the concepts that underpin the studies in this research.

The most fundamental question in SAoT is "How to classify the sentiment of a document?" At the onset, the most basic algorithm that one would design to perform this task would probably be to count the number of positive and negative words that are contained within the text. A decision as to whether the overall text portrays a positive or a negative sentiment would then be made based on the resulting majority. This basic strategy shall be referred to here as the "Naïve Majority algorithm." Although this technique would probably be only slightly better than a random choice baseline of 50%, at least two important questions arise from its simple architecture: How would the sentiment polarity of (a) the words, and (b) the entire document be identified?

These two questions about the Naïve Majority algorithm are closely linked to the problems that are addressed in this research. Consequently, a significant part this chapter is aimed at reviewing the manner in which previous studies have attempted to attempted to answer these questions. The chapter also at highlights some of possible avenues for future work.

2.1 Identifying the Polarity of a Word

Although identifying the sentiment polarity of a word might seem straight forward to the human mind, automating the process might not be quite as simple. Nonetheless, there have been a number of studies that have attempted to address this problem with a reasonable degree of success.

The Literature reveals that these studies can roughly be divided into two main camps: Those that employ either un-supervised [e.g. Hatzivassiloglou and McKeown, 1997; Turney, 2002] or supervised [e.g. Pang et al., 2002] learning algorithms to identify the polarity of words. The main factor that distinguishes these two groups is that only the latter has access to training examples that have been labelled with respect to sentiment. Although these labelled examples give an advantage to supervised algorithms, un-supervised ones have also been quite successful and are essential in situations where the training examples lack sentiment labels.

2.1.1 Un-Supervised Approaches

Most un-supervised approaches to identifying the polarity of a word are based on Part-of-Speech (PoS) pattern filtering [Justeson and Katz, 1995]. This is a simple but yet powerful technique that was adopted from NLP. A precursor to PoS pattern filtering is that the words in the text to be filtered need to be assigned with their corresponding PoS tags. PoS tagging, also called grammatical tagging, is the process of marking up the words in a text as corresponding to a particular PoS, based on both definition as well as context, i.e. its relationship with adjacent and related words in a phrase, sentence, or paragraph. PoS tagging was once performed by hand, but is now done in the context of computational linguistics, using algorithms which associate discrete terms, as well as hidden PoS, in accordance with a set of descriptive tags.

Once the input text has been tagged, a PoS pattern can then be used to filter out words or phrases that have certain properties. For example, a PoS pattern such as "`JJ CC JJ`" can be used to identify opinionated phrases such as "strong and reliable" which consist of an adjective `JJ` a conjunction `CC` and another adjective. Most of the studies described in this section are based on a similar kind of framework.

Conjunctions Constrain Sentiment Orientation

One possible approach to identifying the polarity of a word would be to create an exhaustive reference table that contains all possible positive and negative words. Such an enormous task could be simplified by noting that adjectives tend to be the main source of sentiment rich words. This would therefore reduce the problem to one of identifying the sentiment orientation of adjectives.

Hatzivassiloglou and McKeown [1997] were amongst the earliest pioneers in automating the prediction of the orientation of adjectives. They identified a number of useful heuristics that could aid in the task. For instance they showed that conjunctions such as "and" and "or" almost always constrain the orientations of the two adjectives they connect to be the same. The situation is however reversed for the conjunction "but," which usually connects two adjectives of different orientation. These points are illustrated by the three possible completions of the following sentence:

The tax proposal was

$$\left\{\begin{array}{l} \text{simple } \textit{and} \text{ well-received} \\ \text{simplistic } \textit{but} \text{ well-received} \\ \text{simplistic } \textit{and} \text{ well-received}^{*} \end{array}\right\}$$

by the public

Note that the third completion is ambiguous as the two adjectives "simplistic" and "well-received" posses opposite sentiment orientations.

Hatzivassiloglou and McKeown [1997] utilised this information to develop an algorithm for identifying the orientation of adjectives. Firstly, the algorithm extracts all conjunctions of adjectives from the given corpus. Information from different conjunctions is then combined to determine whether each two connected adjectives are of the same or different orientations. The resulting information is then mapped onto a graph whose nodes represent adjectives, and whose links represent the difference in orientation between the adjectives. A clustering algorithm is then used to separate the adjectives into two clusters of different orientation. This is done by trying to place as many adjectives of the same orientation as possible into the same cluster. Finally, the cluster with the highest average frequency of adjectives is then labelled as positive. This labelling follows from a heuristic that was derived in the same paper, and based on at least two previous studies [Hatzivassiloglou and McKeown, 1995; Lehrer, 1985]. This algorithm proved to be quite effective in classifying adjectives, yielding accuracies ranging from 78% to 92% depending on the amount of available training data.

Using IR to Identify the Orientation of Phrases

Although Hatzivassiloglou and McKeown [1997] fully attained their objective of predicting the orientation of adjectives, more information would still be required in order to serve the current

purpose of predicting the orientation of the larger encompassing text. One problem in particular is that although an adjective almost always expresses an opinion, it also has the potential of being quite misleading if interpreted in isolation. For instance, the adjective "unpredictable" has a negative orientation in the context of "unpredictable steering." It is however positive in the context of "unpredictable plot" [Turney, 2002].

Envisaging this problem, Turney [2002] developed the PMI-IR algorithm which takes into account the context of a word. Hence, rather than computing the orientation of a single adjective, it would compute the orientation of a bi-gram[1] containing two words where one would either be an adjective or an adverb, and the other would provide context. The PMI-IR algorithm is based on both Pointwise Mutual Information (PMI) [Church and Hanks, 1990] and Information Retrieval (IR) [Rijsbergen, 1979]. It compares the co-occurrence between the selected bi-gram and two manually predefined sets of oppositely oriented words. In Turney's study, the positive set contained the word "excellent" whereas the negative one contained the word "poor." If the bi-gram was found to co-occur more frequently with the positive set, then it would be classed as positive; otherwise it would be negative. The co-occurrence between the bi-gram and the two sets was estimated over the World Wide Web via the AltaVista advanced search engine[2]. Turney [2002] chose this search engine because of its **near** operator which constrains the search to documents that contain the query terms within a distance of 10 words in either direction of each other. This technique has been shown to be more effective than the "**and**" operator which simply retrieves all documents containing the query terms [see Turney, 2001].

Possibilities for Future Work

Although the PMI-IR algorithm provided context to isolated adjectives, one aspect that Turney [2002] overlooked was that it may be beneficial to retain at least some of the adjectives in their initial context-free form. A good example of this is when First Person Narratives, such as the adjective "undesirable" in the extract "When asked about the trip he said it was "Undesirable."." Note here that Turney's PoS filters would overlook this particular adjective. Turney [2002] also did not consider the possibility of retaining tri-grams such as "very fast car." Retaining such entities offers the flexibility of estimating the overall orientation of the text from three different sources,

[1] Two consecutive words.

[2] http://www.altavista.com/sites/search/adv

i.e. uni-grams, bi-grams, and tri-grams. Another aspect worth noting is that the PoS patterns that Turney employed were manually crafted. This can be an especially difficult task [see Jackson and Moulinier, 2002;Section 4.2], and a possible alternative could be to automate the process. These highlighted issues have the potential of improving sentiment classification performance and are amongst some of the avenues that shall be explored into greater detail during the course of this research.

2.1.2 Supervised Approaches

One common property of the previously discussed algorithms is that they were not designed to utilise examples that have been labelled with respect to sentiment. They are, therefore, classed as un-supervised techniques that are employed in the event that the training documents lack sentiment labels. However, with the availability of labelled documents, it becomes possible to employ supervised machine learning techniques to perform the same task. Although this additional information enables supervised techniques to generally outperform un-supervised ones, one weakness with them is that they are typically based on non-linguistic principles. They, therefore, do not provide any explicit linguistic explanations as to why the words or phrases that they retrieve should be considered as being rich in sentiment. Any answers would normally have to be interpreted either from a statistical standpoint, or through manually tracing back the words and phrases to their original contexts within the text. Nonetheless, the answers that one finds are often justified and this makes it typically worthwhile to employ supervised techniques wherever possible.

Using Frequency to Identify Word Polarity

Amongst the first landmark studies on supervised techniques in SAoT research was the work done by Pang et al. [2002]. In this study Pang et al. [2002] classified the sentiments in movie reviews by employing supervised techniques to first identify the polarity of the words within the reviews, and then classify the reviews themselves. As a preliminary step to justify the need for supervised techniques, Pang et al. [2002] performed a small experiment whereby they compared the performance of two graduate students against that of a supervised technique. The students were independently asked to choose indicator words for positive and negative sentiments in movie reviews. Their responses were then used to develop decision functions, similar to the Naïve Majority algorithm, that

	Un-Supervised				Supervised	
	Student A		Student B		Stats + Introspection	
	positive:	negative:	positive:	negative:	positive:	negative:
Words	gripping mesmerising riveting spectacular cool awesome thrilling badass excellent moving exciting	bad cliched sucks boring stupid slow	dazzling brilliant phenomenal excellent fantastic	suck terrible awful unwatchable hideous	love wonderful best great superb still beautiful	bad worst stupid waste boring ? !
Accuracy	64%		58%		69%	
Ties	39%		75%		16%	

Table 2.1: Comparison of Supervised and Un-Supervised Techniques [Pang et al., 2002]

would essentially count the number of proposed positive and negative words within a given test document and classify it according to the majority. On the other hand, the supervised technique gathered its list of positive and negative words based on frequency counts performed over a set of positive and negative training documents respectively. The two resultant lists were then trimmed and refined by introspection, after which they were fed into the same decision functions as those of the graduate students in order to compare the performances.

Table 2.1 illustrates the various word lists and their respective performance outcomes. Note that the supervised technique yielded the highest accuracy with the fewest ties. In addition to this, some of the items in its list such as "?" or "still" do not readily spring to mind as being rich in sentiment. Although by tracing them back to their initial contexts, one sees their merit: The question mark tends to occur in sentences like "What was the director thinking?" whereas "still" appears in sentences like "Still, though, it was worth seeing." Pang et al. [2002] concluded from these preliminary experiments that it was worthwhile to explore supervised corpus-based techniques to identify the polarity of words and to perform sentiment classification in general.

After having established the need for supervised techniques, Pang et al. [2002] proceeded to perform their study on a corpus of 700 positive and 700 negative movie review documents. They investigated various different techniques of identifying the polarity of a word some of which may not be entirely relevant to the current context. However, one common aspect about these techniques is that they were all based on the hypothesis that the most frequent word in a sentiment

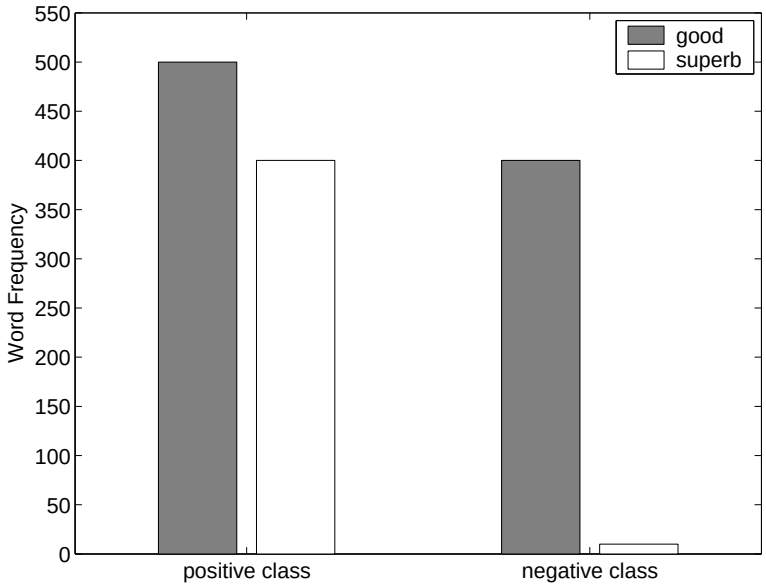

Figure 2.1: The Problem with the Frequency Hypothesis

rich corpus, after stop-words, should also be the richest in sentiment. That is to say that a word would be labelled as positive if it occurred more frequently in positive reviews, and negative if it was more common in negative reviews.

Using Information Theory to Identify Word Polarity

Although the Frequency hypothesis is sufficient in cases where words are relatively uni-polar, it breaks down in scenarios where words occur frequently in both positive and negative reviews. To illustrate this problem more clearly, consider the graph in Figure 2.1 which shows the distribution of the words “good” and “superb” across a positive and a negative class. The word “good” occurs more frequently in the positive class and hence the frequency hypothesis would rank it as being more positive than the word “superb.” Note, however, that “good” also occurs frequently in the negative class unlike “superb” which is almost absent here. This clearly makes “superb” the more positive word of the two but the frequency hypothesis is unable to make this distinction. A solution to this problem requires a technique that not only checks the rate at which a word occurs in a given class, but also the rate at which it is absent in *other* classes.

A good candidate solution to this problem is the Information Gain (IG) measure [Cover and Thomas, 1991]. IG is a measure that is based on Entropy, which is basically a metric of how *surprising* an entity is within its environment. For instance, an entity that occupies 98% of its

environment is not as surprising as the one that occupies the remaining 2%. This is because the former entity is common whereas the latter one is rare. Similarly, the word "good" in Figure 2.1 occurs frequently in both classes and hence is relatively less surprising to observe than the word "superb." IG is particularly good at identifying such anomalies and would immediately rank the word "superb" as being more positive than the word "good." Indeed several previous studies, both in Sentiment Analysis and Text Classification in general [Sebastiani, 2002; Yang and Pedersen, 1997; Mukras, 2004], have considered IG to be representative of the current state-of-the-art in the field of feature selection.

Possibilities for Future Work

Despite the elegant manner in which IG selects features, there still exist a number of SAoT problems that still pose a real challenge to it. One such example is the task of selecting features from a SAoT problem with ordinal classes. The difficulty in accomplishing this task is that IG, as with most feature selection heuristics, is based on the assumption that all pairs of classes are equally similar. This is however not true with an ordinal SAoT problem, in that the similarity between any pair of its classes is a function of the ordinal distance between them. For instance, a textual review accompanied by a rating of 1 (on a 10 point scale) is expected to be more similar to one rated at 2 than another at 10. This consequently means that an important feature in an ordinal SAoT problem is one that is distributed across similar classes. Most feature selection techniques, including IG, cannot easily detect such features principally due to their assumption that all classes are equally similar. Under this assumption, features that occur in similar classes are considered to have an equal importance to those that occur in dissimilar ones, and hence the most appropriate features could easily be overlooked.

One of the main objectives of this research is to develop a solution that will enable IG, and other similar feature selection heuristics, to overcome the assumption of equal inter-class similarities.

2.2 Identifying the Polarity of a Document

As with identifying the polarity of a word, the techniques in identify the polarity of a document are also divided into two groups, i.e. un-supervised and supervised. Note, however, that the majority

of the methods in this area are of the latter kind. This owes mainly to the relative simplicity in assigning a sentiment label to an entire document as opposed to labelling its individual words.

2.2.1 Un-supervised Approaches

According to the Naïve Majority algorithm, the polarity of a document would equate to the sum of the polarities of its individual words. This is probably amongst the first options that one would consider if faced with an un-supervised sentiment classification problem. Indeed, as previously mentioned, Turney [2002] adopts this procedure by summing up the semantic orientations of individual phrases in order to obtain that of the whole document. However, in his conclusion, Turney [2002] also stated that one main drawback with this approach is that "the whole is not necessarily the sum of the parts." This becomes evident if, for instance, a document is excessively sarcastic hence resulting in an overall positive aggregate, while in reality the document should actually be negative.

Exploiting Positional Information

One possible option for avoiding problems such as sarcasm is to exploit the positional information of the phrases within the document. In a previous study that employed supervised learning techniques, Pang and Lee [2004] observed that the end sentences within a movie review yielded a higher performance accuracy than those at the front. Although Pang and Lee [2004] also found that the full review yielded the best overall performance, their study revealed the important fact that the underlying sentiment of a document is not evenly spread along its text. This information could have the potential of improving the performance of un-supervised learning techniques such as Turney's PMI-IR algorithm in that different weights could be assigned to the polarity of a phrase depending on its position within the text. For instance, given the observation by Pang and Lee [2004], one could assign a higher weight to phrases that occur in the rear part a document as they would posses a better description of the overall sentiment. Note, however, that this weight assignment procedure must be performed with care because, as this research will later show, not all domains can be treated in the same way. It may be necessary to first manually label a sample of the documents in the domain and then use supervised techniques, similar to Pang and Lee [2004], to decide on the weighting criteria. This procedure is amongst one the studies that will be

investigated in greater detail during the course of this research.

2.2.2 Supervised Approaches

Supervised Learning techniques have been quite successful in SAoT research. This owes mainly to the availability of training documents, labelled with respect to sentiment, which is a precursor to employing these techniques. Most supervised learning techniques are based on the Vector Space Model [Salton et al., 1975] which is a technique for representing text documents as vectors. Each dimension of a document vector corresponds to a separate feature. If a feature occurs x times in the document, then its value in the vector is x. A collection of such vectors is known as a term-document matrix whose rows and columns conventionally represent documents and terms (i.e. features) respectively. It is important to note that the matrix represents a feature space, and that the vectors of each class tend to reside within separate regions of this space. The task of a supervised learning algorithm would then be to identify the boundaries of these regions by learning them from a sample of the document vectors. This sample is what constitutes the training documents. Once the boundaries are identified, the supervised learning algorithm is said to be trained, and would then be expected to correctly classify the labels of previously unseen documents to a reasonable degree of accuracy.

Ordinal Sentiment Classification

In this research, supervised learning techniques are mainly used to solve SAoT problems with ordinal classes, commonly referred to here as Ordinal Sentiment Classification (OSC) problems. Such problems possess two important characteristics: (1) They are multi-class in nature with three or more classes, and (2) Their classes exhibit similarities that decay with the ordinal distance between them. Some classifiers can inherently solve problems with multiple classes such as OSC, while others can only solve two-class problems. Consequently, for the latter type of classifiers, an alternative technique is to decompose a multi-class problem into a collection of binary ones.

The Literature suggests that OSC has been a relatively unexplored research area. This owes mainly to the relative youth of SAoT in general. As a result of this, there is a general lack of clarity on which classifiers are best for OSC problems. Nonetheless, there are a number of notable pioneering studies that have looked into certain aspects of the problem. The work by Koppel and

Schler [2006], for instance, was quite fundamental in illustrating the need for specialised classifiers to tackle OSC problems. Their basic argument was that most research on Sentiment Analysis, at the time, often ignored "neutral" examples, and only focused on examples of significant polarity, i.e. positive or negative. To prove that neutral examples were important, they performed OSC on three ordinal datasets using a classifier that treats the classes as unordered, and another that respects class ordering. Their results clearly showed an improvement in classifiers that respected class ordering over those that did not.

The study by Pang and Lee [2005] was also based on a similar argument as that of Koppel and Schler [2006]. They, however, went a step further and proposed a purpose designed algorithm called metric labelling. Metric labelling is a meta-classifier that is used in conjunction with a base classifier. Given an m class problem, metric labelling basically maps both the predictions of a classifier, and the similarities between the classified documents onto a graph. It then uses graph theory to partition the graph into m parts and thereby relabelling the documents. In a sense, metric labelling can be thought of as penalising the base classifier for assigning divergent labels to similar documents.

Pang and Lee [2005] evaluated the metric labelling system by comparing its performance against that of its base classifiers. Their results were quite interesting in that, in half of their experiments, the metric labelling system was unable to enhance the performance of the Regression based Support Vector Machine (SVM-reg) base classifier. This has important implications because SVM-reg is more suitable for regression rather than classification. Nonetheless, the nature of OSC problems is that they posses an ordering similar to that in regression and this makes it possible for one to treat them as regression problems as well. Unexpected observations, such as that made by Pang and Lee [2005], offer interesting insights on the possible classifiers for OSC. There is still very little literature on the most effective systems for OSC, and hence one of the objectives of this research is to investigate a variety of classification techniques as applied to OSC problems.

2.3 Summary

This chapter presents a survey of SAoT in light of the studies that are conducted in this research. It identifies a number of techniques that exploit linguistic properties to identify the orientation of words. However, one aspect about many of these systems is that the linguistic rules that they

employ are normally hand crafted. This makes it difficult for them to adapt to changing environments without human intervention, which is a setback in terms of both time and cost. The chapter also identifies a number of feature selection techniques such as IG and Feature frequency. It also highlights their weakness with regard to their ignorance of inter-class similarities, such as those exhibited in OSC problems. Overcoming these limitations constitutes a key motivation of this thesis.

A number of document classification techniques were also looked at including the aggregation of individual word polarities, and the use of standard text classifiers. Again, as with the feature selection techniques, a number document classifiers are also ignorant of inter-class similarities which greatly disadvantages them when addressing OSC problems. Purpose designed classifiers such as the metric labelling algorithm were also discussed. It was however noted that insufficient literature exists in this area which motivates a further contribution of this thesis in terms of providing a thorough evaluation of various classifiers on OSC.

Chapter 3

Background

One distinct aspect about sentiment classification problems is their apparent difficulty relative to standard topic based classification problems. Pang et al. [2002] for instance found that, under similar conditions of uni-grams features, standard text classifiers would perform "at best" 12% lower on sentiment classification than on other topic based problems. A similar observation was also made when Chakraborti et al. [2007] applied *Sprinkled* Latent Semantic Indexing (LSI) to hierarchical, orthogonal, and ordinal text classification problems. Their results clearly indicated significantly lower performances on the ordinal problem, which translates to a special case of sentiment classification with ordered classes. The minimum performance disparity in this case was about 20% accuracy. The argument Chakraborti et al. [2007] gave for this was that the classes were not neatly separable, partly because the ordinal nature of problem inherently results in the use of different ratings to express similar judgements. Similar views on the difficulty of sentiment classification problems have also been reported in the literature and the following list provides a sample of these studies [see Turney, 2002; Mukras, 2004; Pang and Lee, 2005; Koppel and Schler, 2006; Mukras et al., 2007b].

The objective of this chapter is to establish a framework that will enable this research to investigate the reasons behind this relative lacklustre performance on sentiment classification problems. The chapter begins by discussing the datasets used throughout the research. These include at least six different corpora, two of which were custom-compiled for this research. The chapter then goes on to discuss four important issues, namely: (1) The default textual pre-processing procedure, (2) feature selection heuristics, (3) classifiers, and (4) performance metrics.

Type of Dataset	Name of Dataset	Total Classes	Docs per Class	Total Tokens	Mean Doc Length
Bi-Polar	Polarity	2	1000	702,189	351.095
Ordinal	Actors	5	500	61,713	24.685
Ordinal	Edmunds	28	100	113,576	40.563
Ordinal	Scale	8	100	161,404	201.755
Neutral	Reuters	2	1000	158,412	79.206
Query	TrecBlog	n/a	n/a	1,707,345,106	537.728
Lexical	WordNet	n/a	n/a	n/a	n/a

Table 3.1: Information and Statistics about the Datasets

3.1 The Datasets

Table 3.1 summarises the datasets that were used in this research. Most of these datasets were sourced primarily from online opinion portals such as blogs and review sites. The reason for this was that such domains are mainly authored by independent individuals who aim at expressing their views to the world. Consequently, such sources are typically representative of genuine public opinions, in that they are not biased towards the benefit of any company or organisation. This factor makes them an excellent resource for SAoT Research.

3.1.1 The Problem Datasets

In this research the problem datasets where either bi-polar or ordinal in nature. A bi-polar dataset is essentially composed of two classes: sentiment positive and sentiment negative. On the other hand, an ordinal dataset can contain three or more classes that range from an extreme positive, to an extreme negative sentiment. The similarity between any two classes of an ordinal dataset can be inferred from the labels. For example, a textual review accompanied by a rating of 1 (on a 10 point scale) is expected to be more similar to one rated at 2 than another at 10.

It is important to note that both types of datasets are polar in nature, and that the main difference between them is that ordinal datasets contain one or more intermediate classes between the two poles. It is therefore possible to create a bi-polar dataset from an ordinal one by simply dropping the intermediate classes. This approach was used by Mukras et al. [2007a] and has also been adopted here.

The Polarity Dataset

This dataset[1] was compiled by Pang and Lee [2004] and was initially bi-polar in nature. It is based on the movie review domain and is composed of 1000 positive and 1000 negative movie reviews. It was chosen here as it is amongst the most popular benchmark datasets for SAoT research. Another reason for choosing it is that movie reviews have been noted to be amongst the most difficult of several domains for sentiment classification [Turney, 2002].

The Actors, Edmunds, and Scale Datasets

These datasets were initially ordinal in nature, and were respectively compiled by Chakraborti et al. [2007], Mukras et al. [2007b], and Pang and Lee [2005]. Note that the Actors and Edmunds datasets were compiled as part of this research.

The Actors dataset is based on reviews about actors and actresses from the Rateitall.com opinion website. It contains 2500 documents distributed equally amongst 5 ordinal classes. The class labels range from 1 to 5, where 1 is the most negative class, whereas 5 is the most positive. The classes were designed such that the number of reviews that an author can contribute to any one class is limited to a maximum of 15. This was done to avoid the possible bias of any prolific author from dominating the corpus [a similar approach was used by Pang et al., 2002].

The Edmunds dataset is based on consumer reviews about used motor vehicles from the Edmunds.com website. It contains 2800 documents distributed equally amongst 28 ordinal classes. The class labels range from 1 to 28, where 1 is the most negative class, whereas 28 is the most positive.

The Scale dataset was initially based on 5006 movie reviews each containing an ordinal class label in the range of 0 to 10 (where 0 is the most negative class, whereas 10 is the most positive). It was later reconfigured to suit the purpose of this research as follows. Firstly, the labels were used to partition the reviews into 10 ordinal classes, i.e. $\{$0-1, 2-3, $\ldots$, 9-10$\}$. An equal class distribution was then obtained by randomly selecting 100 reviews from each class. The first and last classes were then dropped as they lacked enough documents. This led to a total of 800 documents distributed equally amongst 8 ordinal classes.

The Actors and Edmunds datasets[2] were compiled specifically for this research whereas the

[1] Available at http://www.cs.cornell.edu/People/pabo/movie-review-data/
[2] Available at http://www.comp.rgu.ac.uk/staff/ram/downloads.html

Scale dataset was obtained from literature.

3.1.2 The Support Datasets

The support datasets included the Neutral dataset (used as a control), the Query dataset (used as a background corpus), and the Lexical dataset (used as a reference for lexical information on words). These have been described as follows.

The Neutral Dataset

In this research, a Neutral dataset refers to one that is orthogonal and does not contain sentiment rich information. Orthogonal datasets contain classes that bear no explicit relationships to each other. For example, the topic "sports" can be considered to be orthogonal to the topic "computers." It is worth noting that although bi-polar datasets contain oppositely oriented classes, they are not orthogonal. This is because their polar nature means that there exists a similarity between their two classes, and the degree of this similarity decreases with the averse in orientation between their classes.

The Reuters dataset was the only Neutral corpus that was used in this research. It was formed by randomly selecting 1000 documents from the ACQ and EARN classes of the Reuters-21578 corpus[3] such that each document belongs to at most one class. The ACQ and EARN classes are orthogonal to each other and contain little, if any, sentiment rich information and hence satisfy the neutral dataset requirements. In addition to this, only two classes were chosen as the neutral dataset is to be compared against the polarity dataset which also has two classes (see Chapter 4).

The Query Dataset

The Trec Blog collection [Macdonald and Ounis, 2006] was used as the Query dataset. Its purpose was to act as a background corpus for making inferences about sentiment related usages of given words and phrases (see Chapter 4). It was compiled by the University of Glasgow and the resultant collection that was used here after pre-processing contained 3,175,111 blog posts. A blog post refers to an entry into a personal site that archives the posts in a reverse chronological order. Blogs are typically rich in opinion as they are authored by individuals who aim at expressing their

[3] Available at http://www.daviddlewis.com/resources/testcollections/reuters21578/

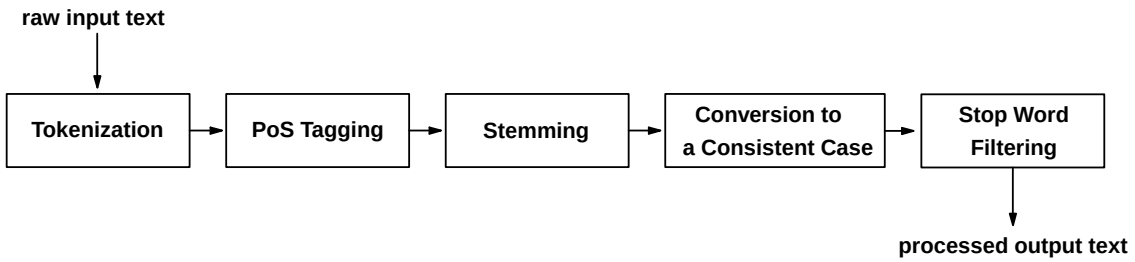

Figure 3.1: The Textual Pre-Processing Pipeline

opinions to the world. The Trec Blog collection was meant to be a realistic 11 week snapshot of the blogsphere (the collective term for all blogs). This makes it an excellent dataset for sentiment related queries.

The Lexical Dataset

WordNet [Fellbaum, 1998] was used as the Lexical dataset. In one sense, WordNet can be thought of as a thesaurus in that its building block is a synset (synonym set) of all words that express a given concept. It, however, goes further than this and links each word in a given synset to other words by various relationships such as hyponymy, meronymy, and entailment.

In this research, one of the uses of WordNet was to expand manually generated sets of words that form *sentiment concepts*. For instance, the set of words {superb, excellent, brilliant} form a concept that describes a strongly positive sentiment. This set can then be expanded by use of WordNet's synset facility. The familiarity score, a WordNet measure of the popularity of a word, can then be used to filter out set members that are either hardly used or too common. WordNet could also, in a similar fashion, be used to generate contrasting sentiment concepts by use of the antonym facility. This feature is particularly useful in un-labelled sentiment analysis whereby oppositely oriented sentiment concepts need to be generated.

3.2 Textual Pre-Processing

Effective text classification is to a degree predicated on the quality of textual pre-processing that has been applied. There are, however, no hard and fast rules about the best pre-processing steps to follow. Each problem is potentially different and hence requires special treatment. Little work has been done to investigate the best pre-processing practises for SAoT and this research aims to

	Stage	Resultant Sentence
1	Initial Sentence	Smoking can lead to impotence.
2	Tokenization	Smoking can lead to impotence .
3	PoS Tagging	Smoking_VVG can_VM lead_VV0 to_II impotence_NN1 ._.
4	Stemming	Smoke_VVG can_VM lead_VV0 to_II impotence_NN1 ._.
5	Convert to single case	smoke_VVG can_VM lead_VV0 to_II impotence_NN1 ._.
6	Stop-Word Removal	smoke_VVG lead_VV0 impotence_NN1

Figure 3.2: The Stages in Textual Pre-Processing

address this problem.

Textual pre-processing is typically performed in several steps as shown in Figure 3.1 and 3.2. First, the raw input text is tokenised so as to divide it into basic words/tokens. Second, the resultant tokens are tagged with their respective PoS tags. This step is normally bypassed for most classification approaches that employ the vector space model. However, Pang et al. [2002] noted that the application of PoS tags could improve the average performance for at least the Naïve Bayes classifier, and as a consequence this study aims to investigate this claim. Third, lemmatisation which is also known as stemming is performed on the PoS tagged tokens. Fourth, the tokens are converted to a consisted case. This avoids the machine from distinguishing between tokens such as “HERE” and “Here.” It is, however, possible that a sentiment classifier may benefit from this distinction in that the emphasis of capitalisation in a token such as “AWESOME,” as compared to “awesome,” would be appreciated. Finally stop-word filtering, which eliminates words that are poor discriminators, is performed.

Each of the steps in Figure 3.1 was empirically evaluated in the General Appendix. The steps that were chosen for this research are as follows: Tokenization is always performed, PoS tagging and Stemming are only performed when specified, Conversion to a consistent case and stopword removal are always performed.

3.3 Feature Selection

A major characteristic of Text classification problems is the high dimensionality of the feature space. A moderate sized collection can easily result in a feature space containing several thousands of dimensions. This is prohibitively high for most machine learning algorithms. Feature Selection therefore aims to reduce the original feature space without sacrificing classification accuracy.

Although most feature selection techniques have been well studied, the sheer number of possible feature combinations for any given collection makes it almost impossible for anyone to conclusively recommend the "best feature selection technique." For this reason, it is preferable to view to feature selection techniques as a heuristics rather than algorithms. Nonetheless, despite the apparent equivocal nature of feature selection, there have been a number of authoritative studies [e.g. Yang and Pedersen, 1997; Sebastiani, 2002; Forman, 2003] that have given useful guidelines on which heuristics to employ. The majority of these studies seem to agree on a decreasing performance trend starting from Information Gain (IG) → Chi-Square Score (CHI) → Document Frequency (DF) → Term Strength (TS). This research employs the first three of these, which are described as follows.

Information Gain

Let f_k be an arbitrary feature in a training dataset whose documents are divided into m mutually exclusive classes, i.e. $\mathcal{C} = \{c_1, \ldots, c_m\}$. Information Gain would then assign a score to f_k as follows:

$$IG(f_k) = \sum_{c \in \mathcal{C}} \sum_{f \in \{f_k, \bar{f}_k\}} P(f, c) \log \frac{P(f, c)}{P(f)P(c)} \tag{3.1}$$

Here $P(c)$, $P(f)$, and $P(f, c)$ are calculated by sums over all documents – that is $P(c)$ is the number of documents with class c divided by the total number of documents; $P(f)$ is the number of documents containing one or more occurences of feature f divided by the total number of documents; and $P(f, c)$ is the number of documents with class label c that also contain word f. IG can be thought of as a test of independence between the variables f and c. This is evident from the sum of logarithms in Equation 3.1 which would yield a value of zero if $P(f, c) = P(f)P(c)$, meaning that f and c are independent, and a value greater than zero otherwise. Consequently, the larger the value of $IG(f_k)$; the more useful the feature f_k would be for classification.

The Chi-Squared Score

The Chi-Squared score is also a test of independence that compares observed frequencies against the expected ones. In order to describe it in the context of feature selection, assume that one wishes to test the independence of feature f_k with respect to occurring in one of the m classes contained

	c_1	...	c_m	Total
f_k	$P(f_k, c_1)$	...	$P(f_k, c_m)$	...
$\bar{f}_k$	$P(\bar{f}_k, c_1)$	...	$P(\bar{f}_k, c_m)$	...
Total	...	...	...	$\sum_{f,c} P(f, c)$

Figure 3.3: A Contingency Table for an m Class Problem

in the set $\mathcal{C} = \{c_1, \ldots, c_m\}$.

In order to do this, one would first set up a null hypothesis that states that the feature f_k is independent of all the m classes. The Chi-Squared Test would then be used to test this hypothesis. The contingency Table in Figure 3.3 illustrates the values that would need to be calculated. Once these values are at hand, the test of independence can then be performed by calculating:

$$\chi^2(f_k) = \sum_{c \in \mathcal{C}} \sum_{f \in \{f_k, \bar{f}_k\}} \frac{[P(f, c) - E(f, c)]^2}{E(f, c)} \tag{3.2}$$

Where $P(f, c)$ is the observed frequency and $E(f, c)$ is the expected frequency asserted by the null hypothesis. $E(f, c)$ is computed as the column total for class c, times the row total for feature f, divided by the grand total $\sum_{f,c} P(f, c)$. Equation 3.2 has a value of zero if f_k is independent of the m classes, and its value grows with the level of dependence.

The Chi-Squared statistic computed in Equation 3.2 has $m - 1$ degrees of freedom (CHI_{m-1}). The reason for this is that the degrees of freedom correspond to the number of total possible outcomes (or classes in this case) minus 1. Note also that it is also possible to work with 1 degree of freedom (CHI_1). In this case the Table in Figure 3.3 would be reduced to a 2×2 Table. Equation 3.2 would also be greatly simplified to the following formulation:

$$\chi^2(f_k, c) = \frac{N \left[P(f_k, c)P(\bar{f}_k, \bar{c}) - P(f_k, \bar{c})P(\bar{f}_k, c)\right]^2}{P(f_k)P(\bar{f}_k)P(c)P(\bar{c})}, \quad c \in \mathcal{C} \tag{3.3}$$

Note that, since the Equation 3.3 only has 1 degree of freedom, it would be necessary to calculate the independence of feature f_k separately for all m classes. These m computations would then have to be aggregated into one global score. It is common practise to take either the average or the maximum of this value these m computations, and the latter technique was employed in this research.

Document Frequency

Finally, Document Frequency is the number of documents within which the feature f_k has been observed. Assuming that $P(f_k)$ is estimated by only recognising the presence of a feature f_k in a document, DF would be computed as follows.

$$DF(f_k) = N \cdot P(f_k) \tag{3.4}$$

Although DF is the simplest of the three techniques, strong correlations have been found between the features selected by DF and the other two techniques [Yang and Pedersen, 1997]. This therefore means that it can be reliably used instead of IG or CHI when the computation of these heuristics is too expensive.

Most of the studies in this research reduce the initial feature space by 98% using Information Gain. These two heuristics have been used in a number of previous studies [Yang and Pedersen, 1997; Wiratunga et al., 2004] and are considered as good rules of thumb.

3.4 The Classifiers

Active research in Sentiment Classification began around the mid 90's. During that time, the main themes were related to the prediction of either the orientation of adjectives [Hatzivassiloglou and McKeown, 1995 1997], or the subjectivity of text [Wiebe, 1994; Wiebe et al., 1999]. However, it was not until its adoption of Text Classification techniques in the late 90's did the field start gaining the popularity that it currently enjoys.

Text classification has been around for much longer and, in some references, Sentiment Classification has been suggested as being one of its subsets [Pang et al., 2002]. Until the late 80's the most popular approach to Text Classification was to manually write a query for each category of interest. The query could then be used to guide a search, and the documents retrieved from the search could then be classified to the corresponding category. A good example of this architecture is the Construe/TIS System which assigns zero or more labels to stories for a Reuters news database [Hayes and Weinstein, 1991]. The core of the program is a set of concept rules that identify key concepts in the text and trigger the assignment of category labels. For example, in order to classify into say the "Australian dollar" category, a data practitioner would design concept rules

that would identify: (a) clear references of "Australian dollar," or (b) references to "Australian" and "dollar" without confounding references to the "US dollar" or the "Singapore dollar."

Although such systems are typically quite effective, the downside to them is the manual cost involved in their development which is clearly evident with the 6.5 person years that it took to complete the Construe/TIS system. With the advent of the 90's, concept rule systems gradually gave way to text classifiers that were based on Machine learning techniques which were significantly cheaper to deploy.

A machine learning text classifier essentially operates by being supplied with a set of training documents $\{x_i\}_{i=1}^{N}$ whose labels $\{y_i\}_{i=1}^{N}$ could either be available (a supervised problem) or unavailable (an un-supervised problems). The documents are typically represented as a vector whose components are features, such as words or phrases occurring in the text. The training documents typically follow some unknown distribution that maps x_i to y_i, and the task of the machine is to use the documents in order to *learn* this distribution. Having done so, the machine is then expected to correctly predict the label of an unseen test document that follows a similar distribution as the training examples.

In this research, two main machine learning classifiers were employed: Naïve Bayes and the Support Vector Machine. These two classifiers are popular in both SAoT [Pang et al., 2002; Wilson et al., 2004] and Text Classification in general [Gabrilovich and Markovitch, 2004; Sebastiani, 2002].

3.4.1 Naïve Bayes

Naïve Bayes, in spite of its name, is a very powerful classifier which is both simple and easy to interpret. It is probabilistic in nature and operates by building statistical models of the classes it assigns to. For instance, the Naïve Bayes described here models the distribution of words in a document as a multinomial.

In order to describe this classifier, assume that the training documents are divided into m mutually exclusive classes, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. The parameters to the multinomial model for class $c \in \mathcal{C}$ would then be: $\vec{\theta_c} = [\theta_{c1}, \theta_{c2}, \ldots, \theta_{cn}]$, where n is the number of features in the vocabulary, $\sum_j \theta_{cj} = 1$, and θ_{cj} is the conditional probability that feature j occurs in class c. The probability θ_{cj} is normally smoothed by a Laplace count in order to avoid it from being zero. The label of an unseen test document $d = [d_1, d_2, \ldots, d_n]$, where d_j is the frequency of feature j in

document d, is then predicted using the Bayes rule,

$$label(d) = \underset{c}{\operatorname{argmax}} \left[P(c) \frac{P(d|c)}{P(d)} \right] \tag{3.5}$$

The probability $P(d|c)$ is estimated by using a multinomial distribution, i.e.

$$P(d|c) = \binom{\sum_j d_j}{d_1, d_2, \ldots, d_n} \prod_j (\theta_{cj})^{d_j} \tag{3.6}$$

The multinomial distribution assumes that the n features in document d are independent of each other. This incorrect supposition is known as the Naïve Bayes Assumption and only holds because of the stochastic nature in which words are used in language [Domingos and Pazzani, 1996].

The multinomial coefficients in Equation 3.6 can be dropped off as they are constant across all classes. Similarly, the probability $P(d)$ in Equation 3.5, can also be dropped off for the same reasons. This simplifies Equation 3.5 to the following formulation:

$$label(d) = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \left[P(c) \prod_j (\theta_{cj})^{d_j} \right] \tag{3.7}$$

The multiple products in Equation 3.7 would most probably lead to an arithmetic underflow and thus it is a common practise to represent it in logarithm space:

$$label(d) = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \left[\log P(c) + \sum_j d_j \log \theta_{cj} \right] \tag{3.8}$$

The label of document d is then taken as the class that yields the maximum value of the resultant Bayes rule formulation as shown in Equation 3.8.

Naïve trains and classifies in linear time with respect to the number of classes. This is explained by noting that, during training, Naïve Bayes computes $|\mathcal{C}|n$ conditional probabilities (i.e. those contained in $\vec{\theta}_{c_1}, \ldots, \vec{\theta}_{c_m}$) and hence trains in $O(|\mathcal{C}|n) = O(mn) = O(m)$ time. Note that n is the number of features and is taken as a constant. Similarly, during classification, Naïve Bayes makes $|\mathcal{C}| + |\mathcal{C}|n$ calculations and hence classifies in $O(|\mathcal{C}| + |\mathcal{C}|n) = O(m + mn) = O(m)$ time.

The simplicity and robustness of the Naïve Bayes classifier has led it to become amongst the most popular of classifiers. It is a favourite choice for industrial applications as it is quite fast and

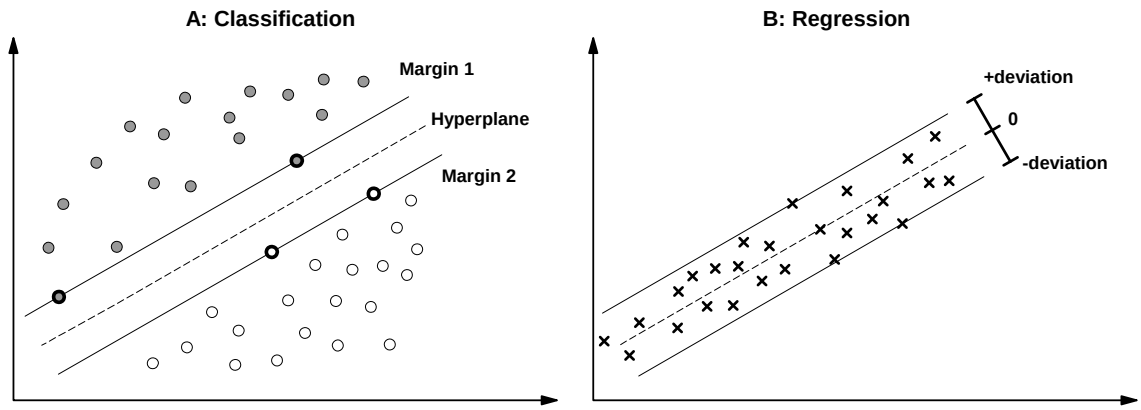

Figure 3.4: Support Vector Machines: Classification and Regression

its performance remains relatively stable under various conditions compared to other classifiers [see Mukras, 2004].

3.4.2 The Support Vector Machine

The Support Vector Machine (SVM) belongs to a family of classifiers that perform classification by building a separating boundary between the classes of interest. A special property of the SVM is that it simultaneously tries to minimise the generalisation error while maximising the geometric margin between the classes. For this reason, it is also known as the *maximum margin classifier*.

The intuition behind SVMs can be explained by Figure 3.4A which illustrates a simplified version of a linear SVM that has been trained on examples from two classes. Here the SVM constructs a separating hyperplane and then tries to maximise the "margin" between the two classes. To calculate the margin, the SVM constructs two parallel hyperplanes, one on each side of the initial one. These hyperplanes are then "pushed" perpendicularly away from each other until they come in contact with the closest examples from either class. These examples are known as the *support vectors* and are illustrated in bold in Figure 3.4A. Intuitively, the best separation is the one with the largest margin between the two hyperplanes. The hope in this is that, the larger the margin; the lower the generalisation error.

SVMs can also be used for regression (SVM-reg). The intuition behind SVM-reg can be explained by Figure 3.4B which illustrates a simplified version of a linear SVM-reg fit over a set of training data points $\{(x_i, y_i)\}_{i=1}^{N}$. Here the goal of SVM-reg is to find a function $f(x_i)$ that has a predefined *maximum threshold* from the actual targets y_i for all the training data points, and

at the same time, is as flat as possible. In other words, deviations between $f(x_i)$ and y_i can be tolerated, so long as they are less than the predefined threshold value.

The Support Vector Machine has been well studied and is regarded by many as the current state-of-the-art in text classification [Joachims, 1998; Gabrilovich and Markovitch, 2004]. It has at least three properties that make it well suited for the text domain. Firstly, SVMs are good at handling the problem of high dimensionality which is a common characteristic of the text domain. Secondly, most text classification problems are linearly separable and the SVM classifier is especially designed for this kind of problem [Joachims, 1998]. Finally, the vectors in a term-document matrix are typically sparse with very few non-zero entries and Kivinen et al. [1997] provided both theoretical and empirical evidence that SVM-like classifiers are well suited for such problems.

The details behind SVMs are complex and certainly beyond the scope of this study. Nonetheless, there are a few SVM implementations that are publicly available and amongst the most common ones is SVM^{light} written in c by Joachims [1998]. All SVM results in this research are based on this implementation[4].

3.5 Performance Metrics

Although sentiment classification is, as is self explanatory, a classification task, it is still possible to approach it as a regression problem. The reason for this is that the classes in a sentiment classification problem are typically ordered. Consequently, inter-class similarity tends to be related to this ordering in that, the closer the classes in the ordering; the more similar they are. These varying inter-class similarities are clearly visible in the 5 graphs of Figure 3.5. Here each graph illustrates the Euclidean distance between the centroid of class c, where $c \in \{1, 2, \ldots, 5\}$, and those of all other classes. Here one can see, for instance in the second graph from the left, that the Euclidean distance between the centroid of class 2 is closer to that of class 3 than to that of class 4.

Such varying inter-class similarities would therefore warrant larger penalties to classifier predictions that *regress* further away from the true class labels. The work by Chakraborti et al. [2007] on confusion matrices was particularly insightful in illustrating the need for unequal misclassification penalties. They found that classifiers experienced more confusion in deciding between adjacent classes of an ordinal scale. Consequently, by assigning an equal cost to all mis-

[4]Available at http://svmlight.joachims.org/

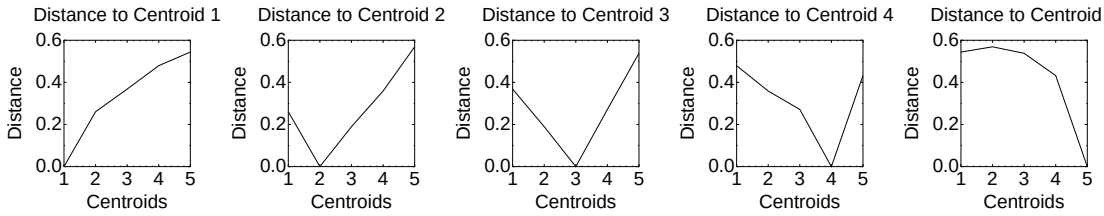

Figure 3.5: Inter-Class Euclidean Distance for each class of the Actors Dataset

classifications, they were essentially discarding all the "nearly correct" predictions made by the classifier.

3.5.1 Mean Squared Error

One option of capturing these "nearly correct" predictions is to employ Mean Squared Error (MSE), which is a commonly used measure of regression. Assuming that $\{y_i\}_{i=1}^{N}$ are the true class labels, and $\{\hat{y}_i\}_{i=1}^{N}$ are the classifiers predictions, MSE would be computed as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3.9}$$

Unlike accuracy, which only registers exact predictions, the MSE tells us how far the predictions regress from the true class labels. It assigns a smaller penalty to a closer prediction and a larger penalty to a further one. This makes it an excellent performance measure for the sentiment classification domain and thus was adopted as the default measure in this research.

3.5.2 Comparing Performance

In this research, the difference in performance between two algorithms was mostly compared using two steps. Firstly, twenty paired samples of performances of the algorithms would be generated using 20 fold cross validation. The statistical difference between these two samples would then be compared using the two tailed t-test at the $p < 0.05$ significance level. A further discussion on this topic has been provided in the General Appendix.

3.6 Summary

The outcomes obtained from the discussions and experiments that were conducted in this chapter form the basis of the default parameters that are used throughout this thesis. The following summarises this information.

The Polarity dataset was the only bi-polar dataset, the Ordinal datasets were Actors, Edmunds and Scale, the Neutral dataset was Reuters, the Query dataset was the Trec Blog Collection, and WordNet was the Lexical dataset. Note also that the Actors and Edmunds datasets were compiled specifically for this research whereas the others were obtained from literature.

For textual pre-processing: Tokenization was compulsory in all experiments, PoS Tagging and Stemming were only performed when specified, and finally Conversion to a consisted case and stop-word filtering were always performed.

Two classifiers are used in this research: Naïve Bayes and the Support Vector Machines. Naïve Bayes performs classification by employing word-to-class conditional probabilities, whereas SVM builds a hyperplane between the two classes. Three feature selection heuristics are employed: Information Gain, the Chi-Squared score, and Document Frequency. The default in this research was to reduce the initial feature space by 98% using Information Gain. These two combined heuristics have been widely used and are considered as good rules of thumb [Yang and Pedersen, 1997; Wiratunga et al., 2004].

The default measure of performance is the Mean Squared Error (MSE). This was chosen as it tells us how far the predictions are from the true class labels. This is important as most of the classes in the datasets used here possess misclassification costs that increase with the magnitude of disparity between the predictions and the true class labels. The two-tailed t-test, at the $p < 0.05$ level, was used to test whether two sets of paired samples are different from each other. Most of the tests employed 20 paired samples that were mostly obtained from 20-fold cross validation.

Chapter 4

Sentiment Extraction with PoS Pattern Analysis

Part-of-Speech (PoS) patterns are sets of consecutive PoS tags that can be used to filter out phrases from a given input text [Justeson and Katz, 1995]. The patterns are typically chosen such that they filter out phrases that conform to a desired linguistic property. For instance, a pattern formed from an adjective followed by a singular noun (`JJ NN1`) would extract sentiment rich phrases such as "fast car," "great person," or "evil motive." The linguistic property in this case is that the first word evaluates the second hence yielding an opinion rich phrase. Other similar PoS pattern combinations are also possible, and it is not difficult to see the potential that this approach could offer, especially if the patterns are well designed.

A good example of a study that employs PoS patterns for sentiment classification is the work done by Turney [2002]. In this study, Turney addresses the problem of un-supervised sentiment classification by use of phrases extracted by manually crafted PoS patterns. Table 4.1 lists a similar set of PoS patterns as those that he used in his study (The disparity is due to a difference is tagsets). In this Table, `J` refers to adjective forms (`JJ`, `JJT`, or `JJR`), `NN1` and `NN2` to singular and plural nouns respectively, `R` to adverb forms (`RR`, `RG`, `RGA`, or `RGR`), and `VV0` to verb forms. To describe the patterns in Table 4.1, consider the fourth one which means that two consecutive words are extracted if the first is a noun and the second is an adjective, but the third (not extracted) cannot be a noun. The third word is checked so as to avoid extracting a bi-gram such as "very fast" instead of "fast car" from the initial phrase "very fast car." Note also that Turney [2002] used phrases instead

	Tag 1	Tag 2	Tag 3 (Not Extracted)
1.	J	NN1 or NN2	anything
2.	R	J	not NN1 or NN2
3.	J	J	not NN1 or NN2
4.	NN1 or NN2	J	not NN1 or NN2
5.	R	VV0	anything

Table 4.1: Manually Selected PoS Patterns

of single words. The reason behind this is that phrases inherently preserve context. For instance, "very good" and "not good" clearly posses opposing polarities and this information would be lost if the two preceding words, "very" and "not," were discarded.

An obvious drawback of using manually selected PoS patterns is that they need to be created by a domain expert in the first place. Furthermore, as anomalies discussed in the previous paragraph illustrate, this can be quite a demanding task even for skilled practitioner. As an estimate, each domain requires up to twenty PoS patterns and personal experience shows that it is not uncommon for each pattern to take up to five hours of skilled labour to design [see also Jackson and Moulinier, 2002;Section 4.2]. This can be a major setback, especially in industries such as marketing or the media where the data continuously changes.

This chapter attempts to address the problem of manual pattern construction by proposing an algorithm that automatically generates a set of PoS patterns for extracting sentiment rich phrases. The algorithm, named the PoS Pattern Selector (PPS) and developed in this research, makes use of a sentiment rich background dataset to learn the set of PoS patterns. Each word in the background dataset is replaced with its respective PoS tag after which standard feature selection heuristics such as Information Gain (IG), Document Frequency (DF), and the Chi-Squared (CHI) score [Sebastiani, 2002; Yang and Pedersen, 1997] are applied to select the top discriminative patterns of lengths one, two, and three. The hypothesis behind this is that patterns that are predictive of a particular orientation, should also extract phrases that are predictive of the same. The results were, however, contrary to this in that IG and CHI, which are traditionally known to yield more predictive features than DF, resulted in the lowest performances [see also Mukras et al., 2007a]. This led to the conclusion that the most effective PoS patterns are those that occur frequently across documents.

Prior to proceeding with the chapter, there is an issue of principle regarding the deployment

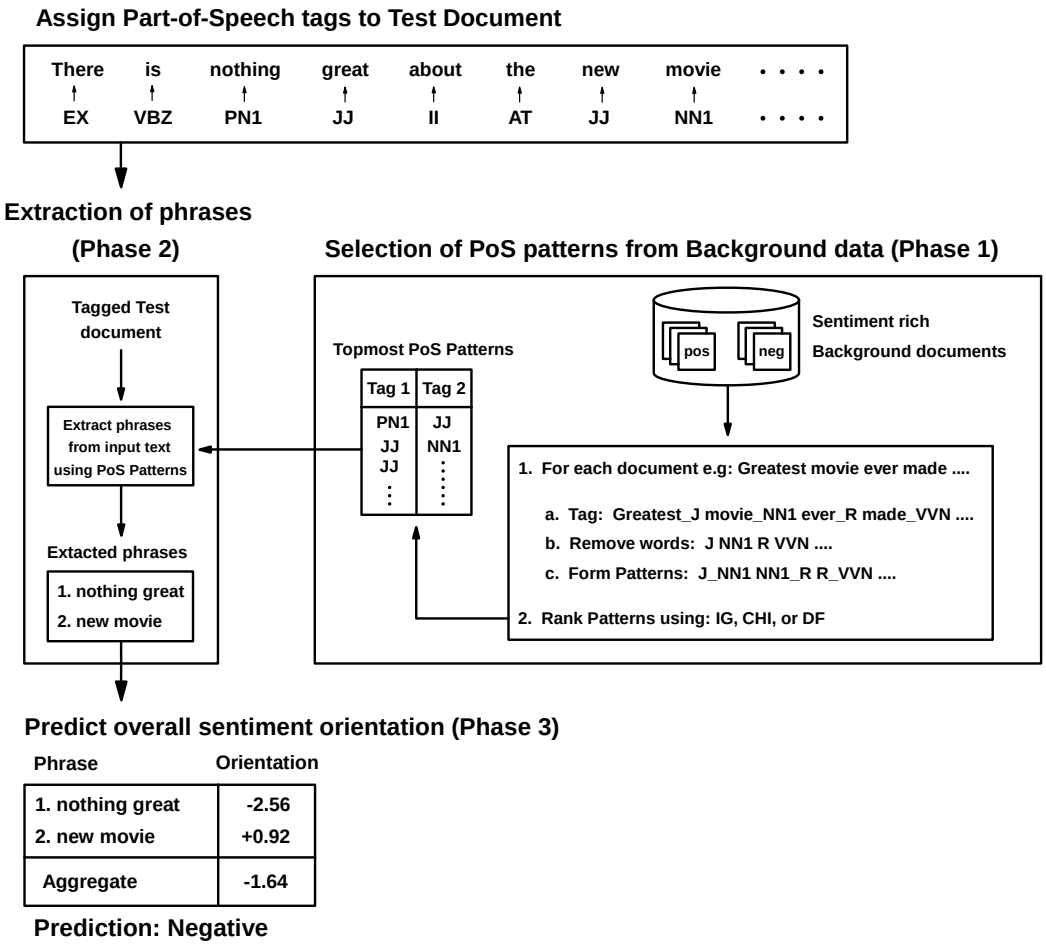

Figure 4.1: The PPS Algorithm in a Sentiment Classification Framework

of background examples on a non-labelled problem, such as this one, that is worth noting here. In such a scenario, there are at least two situations that could arise: The background examples could either be labelled, or non-labelled. There is typically no restriction on applying non-labelled background examples to a non-labelled problem. However, it may be possible to criticise the application of labelled background examples to a non-labelled problem for the reason that it goes against the principle of learning from non-labelled examples. One could, however, defend against this by arguing that although the background examples are labelled, there would still exist some uncertainty as to whether they follow the same distribution as the non-labelled training examples. This type of uncertainty is typically absent when learning from labelled examples. Consequently, so long as this uncertainty exists, then the problem continues to be one of learning from non-labelled examples.

Heuristic	Tag 1	Tag 2	Examples of extracted phrases
IG	NN1	?	shrug ?, glory ?, loser ?
	NN1	!	joke !, understatement !, menace !, perfection !
	VVZ	NN2	grate nerve, play scene, think woman
	NN2	NN2	work life, year job, concert movie
	J	VVG	good look, serious think, good fall
CHI	NN1	J	guy worst, personality decent, spelling unattractive
	NN1	?	shrug ?, moron ?, fear ?, glory ?, loser ?
	NN1	!	joke !, understatement !, menace !, perfection !
	NN1	VVZ	planet act, man pray, character play
	VV0	NN1	walk sunset, get sitcom, show emotion
DF	J	NN1	worst actor, terrible actress, worst breakfast
	NN1	NN1	example non-talent, quality style, courage range
	NN1	J	guy worst, personality decent, spelling unattractive
	NN1	NN2	going look, education work, world affair
	J	NN2	cute star, decent performance, outspoken topic

Table 4.2: Sample of top ranked PoS patterns selected using IG CHI and DF

The remainder of this chapter is organised as follows. Section 4.1 describes the manner in which the PPS algorithm automatically identifies PoS patterns. This section also walks through the steps of how the resultant PoS patterns are used to classify the orientation of an arbitrary test document. The chapter is then concluded in Section 4.2 with a discussion and summary. Note that the empirical evaluation of the PPS algorithm is discussed in the Evaluation chapter 8.

4.1 Overview of the System

The diagram in Figure 4.1 provides an overview of the system that is described by this chapter. As illustrated, it is divided into three separate phases: The first phase, which constitutes the contribution of this chapter, involves PoS Pattern Selection, the second phase deals with the Extraction of Phrases, and the last phase is Un-supervised Sentiment Classification. In order to describe the system, this chapter walks through each of its steps. Note that PoS pattern selection is only performed once, but it needs to be done prior to the other two steps. This will therefore be the starting point of the discussion.

EXTRACT-PHRASES(d, Q)
1. $B \leftarrow \{\}$
2. $M, m \leftarrow |$LONGEST-AND-SHORTEST-PATTERN$(Q)|$
3. **for** $i = 1$ **to** $|d|$ **do**
4. **for** $j = m$ **to** M **do**
5. **if** $i + j \leq |d|$ **then**
6. $t =$ PART-OF-SPEECH$(w_i \ldots w_{i+j})$
7. **if** $t \in Q$ and $w_i \ldots w_{i+j} \notin B$ **then**
8. $B \leftarrow B \cup \{w_i \ldots w_{i+j}\}$
9. **return** B

Figure 4.2: Extraction of Phrases from the Test Document d

4.1.1 Pattern Selection from a Background Dataset

The first step in the PPS algorithm is to establish a background dataset with two classes: sentiment positive and sentiment negative. This dataset is fundamental to the algorithm as its underlying structure implicitly contains the patterns that are required for extracting sentiment rich phrases. Once the dataset is in place, each of its documents are processed so that all words are *replaced* by their corresponding PoS tags[1]. Assuming that $t_1, t_2, \ldots, t_M$ is a sequence of PoS tags in an arbitrary document of this dataset, a PoS pattern of length j would be defined as $t_m \ldots t_{m+j}$ where $m = 1, 2, \ldots, M-j$. All such patterns are then ranked using IG, CHI, or DF.

Once the PoS patterns have been ranked, they are then checked for singular or plural proper nouns (i.e. `NP1` or `NP2`) and those containing any instances of these are discarded. The reason for this is that proper nouns tend to occur within positive, negative, or even neutral phrases. For instance, the proper noun "London" can be found within contexts such as "I love London," "I hate London," and "I don't mind London." This makes it difficult to classify the orientation of a proper noun as its various contexts can be highly ambiguous [Turney, 2002].

4.1.2 Extraction of Phrases

Once the PoS patterns have been processed, the top ranked patterns are then used to extract phrases from the *test document*. The procedure is summarised in Figure 4.2. Here d is the test document consisting of all its words $w_1, \ldots, w_{|d|}$, and Q is a set of PoS patterns of various lengths. In addition to this, the statement $M, m \leftarrow |$LONGEST-AND-SHORTEST-PATTERN$(Q)|$ respectively

[1]Tagging was performed using the RASP system [Briscoe and Carroll, May 2002] which utilises the CLAWS2 Tagset.

Adjective Word	Corresponding Antonyms
new (11,1268194)	old (8,354828), used (3,3)
good (21,719768)	awful (6,29714), terrible (4,38042), bad (14,409)
general (6,574866)	special (7,195450)
right (14,549695)	wrong (9,180121), erroneous (1,2660)
great (6,514301)	terrible (4,38042), ordinary (2,28635)
big (13,410606)	small (10,248872), little (8,505147)
simple (7,245606)	complex (1,44198), difficult (2,77048)
poor (6,113213)	rich (12,74127)
huge (1,109800)	small (10,248872), little (8,505147)
glad (4,103213)	sad (3,82949), bittersweet (2,4273)
smart (7,86815)	stupid (3,104053), weak (12,28502)
foolish (2,10510)	wise (4,32497), all-knowing (1,0)

Table 4.3: A Sample of the Adjectives used to Generate the Sets $\mathcal{P}$ and $\mathcal{N}$

assigns the lengths of the longest and shortest patterns in Q to M and m, whereas the function PART-OF-SPEECH$(w_i \dots w_{i+j})$ returns the PoS tags for the words $w_i \dots w_{i+j}$. The procedure returns the set B which contains all phrases in document d that match the patterns in set Q.

Table 4.2 illustrates, for each feature selection heuristic, a sample of the topmost PoS patterns that were returned, along with a few of the phrases that they extracted. Note that phrases extracted using DF such as "worst actor," and "terrible breakfast" are relatively more intuitive, in terms of sentiment richness, than those extracted by IG and CHI such as "shrug ?" and "moron ?." It will later be shown that classification performances also tend to follow the same trend.

4.1.3 Un-supervised Sentiment Classification

Once the sentiment rich phrases have been identified in the test document, the next step is to compute their respective sentiment orientations. Let b_i be the i^{th} extracted phrase from the test document d. The sentiment orientation of the phrase b_i is computed by comparing its association to a set of positive words $\mathcal{P}$, against its association to a set of negative words $\mathcal{N}$ [Turney, 2002]. The words in these two sets are normally based on antonym pairs. For example, given an entry "good" in set $\mathcal{P}$, there would be a corresponding antonym such as "bad" in set $\mathcal{N}$. The candidate words were also filtered out by querying them against the Query Dataset and selecting the best option. The following describes how the two sets, $\mathcal{P}$ and $\mathcal{N}$, were obtained in this thesis.

The two sets $\mathcal{P}$ and $\mathcal{N}$ were initially compiled from a list of manually selected adjectives as adjectives are known to be good carriers of sentiment [Hatzivassiloglou and Wiebe, 2000].

The words were then recorded onto a list after which a number of sources (including WordNet, thesauri, and plain intuition) were used to generate a list of corresponding antonyms for each word in this list. The two columns in Table 4.3 depict an example of the words that were compiled. This Table also illustrates, for recorded each word, both: (1) Its familiarity-score as an adjective, and (2) Its Query dataset frequency which refers to its frequency in the Trec-Blog Collection [see Macdonald and Ounis, 2006]. These two values are respectively shown in the brackets that follow each word in Table 4.3. The familiarity score was obtained from WordNet [Fellbaum, 1998], and it is a measure of a words usage in normal language. Wordnet computes this score based on the words polysemy count. A word is judged to be polysemous if it has two or more senses whose meanings are related. Consequently, a words polysemy count is the number of related meaning that the word exhibits. The reasoning behind using the polyseme count is that a high score would imply that the word has several adjective polysemes and hence is commonly used, whereas a low score would imply few adjective polysemes and hence is uncommon. Note that the lowest possible familiarity score is 1, whereas there is no upper bound to the number of polysemes that a word can have.

Both the familiarity score and the Query dataset frequency are important parameters in selecting the words that form the sets $\mathcal{P}$ and $\mathcal{N}$, as computing association is difficult with words that are either uncommon or excessively common. For instance, the word "good" in Table 4.3 occurs too frequently. It would therefore not be wise to select it as a member of positive seed word as it would most probably occur in both positive and negative contexts. Similarly, the word "used" occurs too infrequently to be present in any context, let alone a positive or a negative one. The two sets $\mathcal{P}$ and $\mathcal{N}$ were finally chosen as follows $\mathcal{P}$ = {glad, rich, smart, great, wise, huge}, and $\mathcal{N}$ = {sad, poor, stupid, terrible, foolish, little}.

The association between two entities, say x and y, is computed using Pointwise Mutual Information [Church and Hanks, 1990] defined as

$$I(x,y) = \log\left[\frac{P(x,y)}{P(x)P(y)}\right] \tag{4.1}$$

As with IG, Pointwise Mutual Information can also be thought of as a measure of independence in that $I(x,y) = 0$ if and only if x and y are independent of each other, and its value increases with the association between the two. This is easy to see because if x and y are independent, then

SENTIMENT-ORIENTATION$(d, Q, \mathcal{P}, \mathcal{N})$
1. $DO = 0$
2. $B \leftarrow$ EXTRACT-PHRASES(d, Q)
3. **for** each $b \in B$ **do**
4. $\quad DO+ = PO(b, \mathcal{P}, \mathcal{N})$
5. **return** $sign\,[DO]$

Figure 4.3: The Un-Supervised Sentiment Orientation Algorithm

$P(x, y) = P(x)P(y)$, and therefore Equation 4.1 will yield a value of zero; otherwise its value would increase with the association between the two.

This idea can be used to compute the sentiment orientation of the phrase b_i by comparing its association with the two sets $\mathcal{P}$ and $\mathcal{N}$ as follows:

$$PO(b_i, \mathcal{P}, \mathcal{N}) = I(b_i, \mathcal{P}) - I(b_i, \mathcal{N}) = \log \left[\frac{P(b_i, \mathcal{P})P(\mathcal{N})}{P(b_i, \mathcal{N})P(\mathcal{P})} \right] \tag{4.2}$$

Note that if b_i is equally associated to both $\mathcal{P}$ and $\mathcal{N}$, then $PO(b_i, \mathcal{P}, \mathcal{N})$ would yield a value of zero. However, if b_i is more associated to either $\mathcal{P}$ or $\mathcal{N}$, then the value of $PO(b_i, \mathcal{P}, \mathcal{N})$ would either be positive or negative respectively. The probabilities in Equation 4.2 were estimated by using the number of hits returned by a search engine indexed over the Query dataset. This technique was introduced by Turney [2002] and is performed as follows:

$$P(\mathcal{P}) \simeq hits\,(\text{glad} \vee \ldots \vee \text{huge}) \tag{4.3}$$

$$P(\mathcal{N}) \simeq hits\,(\text{sad} \vee \ldots \vee \text{little}) \tag{4.4}$$

$$P(b_i, \mathcal{P}) \simeq hits\,(b_i \ \mathbf{near}\ (\text{glad} \vee \ldots \vee \text{huge})) \tag{4.5}$$

$$P(b_i, \mathcal{N}) \simeq hits\,(b_i \ \mathbf{near}\ (\text{sad} \vee \ldots \vee \text{little})) \tag{4.6}$$

Here $hits(\cdot)$ is a function that returns the number of documents that satisfy its query parameter, and **near** is a binary operator that constrains the search to documents containing its two query parameters, within 10 words of each other in any order. The value 10 was chosen as it is the approximate length of a single sentence. These probability estimates form the basis of the un-supervised classification technique. They determine the rightful orientation of the phrase b_i as per the evidence present in the Query dataset. It is therefore crucial to employ a sentiment rich Query dataset that can generate accurate probability estimates. Finally, the overall document orientation

is then computed as follows:

$$DO(d) = sign\left[\sum_{i} PO(b_i, \mathcal{P}, \mathcal{N})\right] \tag{4.7}$$

A positive aggregate would imply a positive orientation whereas a negative aggregate would imply a negative orientation. The algorithm is summarised in Figure 4.3. The procedure first initialises the orientation of the test document to zero. It then extracts all phrases that match the PoS patterns contained in the set Q and places them into the set B. The orientation of each phrase in the set B is then computed after which the orientation of the sum of all phrases is then returned as the orientation of the overall document.

4.2 Discussion and Summary

This chapter presents the PPS algorithm which is a novel approach to PoS pattern selection for un-supervised sentiment classification. The PPS algorithm automatically mines PoS patterns and hence offers an important alternative to the common approach in literature which involves manual design of the PoS patterns.

The PPS algorithm is divided into three main steps: The first step involves the selection of PoS patterns from a sentiment rich background dataset. This involves tagging the words in the background dataset, omitting the words and retaining the PoS tags, grouping the PoS tags into patterns, and finally ranking the resultant patterns by use of IG, CHI, or DF. The second step involves the extraction of phrases from the test document and the last step classifies the test document based on the extracted phrases.

The evaluation is presented in chapter 8 and involves three main experiments. The first examines the effectiveness of the three ranking criterion (IG, CHI, and DF), the second compares PPS algorithm against manually crafted patterns, and the last examines the effect of a sentiment rich background dataset.

Chapter 5

Positional Information

Despite the fact that positional information has been widely used in topic identification, only a handful of studies in the sentiment analysis domain have attempted to exploit this resource. A module that can propose the most likely locations of sentiment rich phrases within a document would be quite useful to an un-supervised sentiment classifier. Phrases that occur in these locations could be weighted in preference of others and this has the potential of improving classification performance.

The work by Edmundson [1969] was probably amongst the earliest studies to utilise positional information in the domain of topic identification. He stated that "topic sentences tend to occur very early or very late in a document and its paragraphs." This suggests that introductions and conclusions are the most probable locations to identify the topic. In another closely related study, Baxendale [1958] observed a partly similar trend to that proposed by Edmundson. He found that in 85 percent of paragraphs, the topic sentence was in the initial sentence and in 7 percent in the final one. Although this observation was at paragraph level, it clearly suggests that the beginnings or endings of a text are likely positions for topic content. However, there have also been a number of reports in the literature that have had alternative views about the importance of an introduction and/or a conclusion in topic identification. Donlan [1980], for instance, maintained that the topic of a given text could be found anywhere within text or not be mentioned at all. Paijmans [1994] from the Information Retrieval community also found that important terms, as measured using the $tf.idf$ weighting scheme, do not cluster themselves in the first or last sentences of paragraphs. He also mentioned that positional information of this nature is not valuable enough to be considered

in the preparation of indices for Information Retrieval.

In the sentiment analysis domain, the study by Beineke et al. [2004] provides useful insights into how positional information has progressed in this field. They report that professional review summarisers preferred to compile their summaries from quotations that originate from either the early part or the final part of the full review. Their results indicated that the summarisers were particularly fond of the last 5% of the full review indicating that conclusions may be the most indicative of sentiment. Pang and Lee [2004] also observed a similar trend when they found that the rearmost sentences in a review yielded a higher performance than those at the beginning. This gives further evidence in support of the hypothesis that conclusions are typically rich in sentiment, and also that positional information can be a potential indicator of word polarity.

5.1 The Effects of Positional Information

The objective of this section is to examine the importance of positional information and whether it can be incorporated into the sentiment classification procedure. The current hypothesis is that it could be used to guide feature selection such that features that are found within sentiment rich sections of a document could be weighted in preference over others. In order to investigate this possibility, the distribution of sentiment rich words across a document was examined in an attempt to find any existing trends that could be exploited. The *sentiment richness* of a word was estimated as its Information Gain score: The higher a words IG score; the richer its sentiment content. This estimate is justified because it selects words that are most descriptive of the sentiment categories.

The examination was conducted using the standard Actors and Edmunds datasets. Each document, in the two datasets, was split into 20 equally sized portions along its text. The distribution of the top 2% words, as ranked by IG, across the 20 demarcated portions was then recorded. This process was then repeated for 20 folds of cross-validation and the resultant average distribution was mapped onto the graph shown on Figure 5.1.

Note in Figure 5.1 that sentiment rich words tend to lie mainly in the early parts of documents within the Actors dataset. This supports previous observations made by Edmundson [1969] who stated that topic sentences “tend to occur very early or very late in a document.” Beineke et al.’s observations were also partly similar to this in that they found professional summarisers to have a preference to quotes originating from either the front or rear parts of sentiment rich reviews. Note,

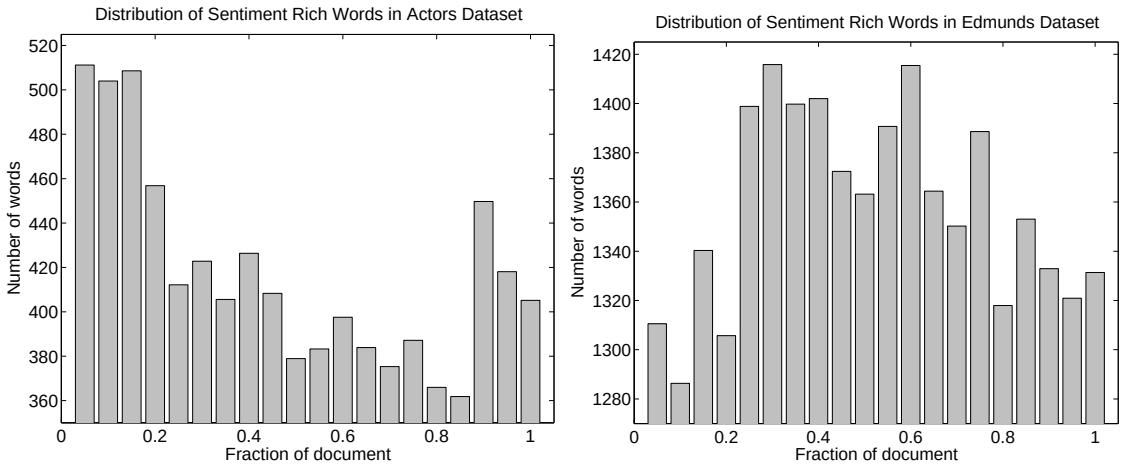

Figure 5.1: Mean Distributions of Sentiment Rich Words across a Document

however in Figure 5.1, that the Edmunds dataset contradicts the observations made in the Actors dataset in that its distribution suggests that the *central body* of a document is the main source of sentiment rich words. This seems to suggest that different corpora may have different distributions of sentiment rich words along their respective documents.

In order to ascertain whether these distributions actually translate into performance, a second experiment that looks into the effect of document structure on mean-squared-error performance was conducted. The experiment investigated four different sections of a document: the introduction (intro), conclusion (con), introduction with conclusion (intro+con), and the body (body). The experiment involved training a classifier on 20 different percentages of a section, and then testing it on full documents each time. The percentages were: 5%, 10%, 15%, . . . , 100%.

Figure 5.2 illustrates the average MSE results that were obtained over 20 folds of cross validation using SVM-reg. Note that the results are mostly in line with the observations made in the previous examination. In the Actors dataset, both the intro and intro+con perform best for most of the percentages that were used. This conforms to the results in Figure 5.1 whereby sentiment rich words in the Actors dataset were distributed mostly in the front half with a few spikes in the conclusions. As for the Edmunds dataset, the body performed poorest during its first 15%. It however picked up to be the best performing section from that point on, and this is roughly in line with its distribution of sentiment rich words as shown in Figure 5.1.

The conformance between the distributions in Figure 5.1 and the mean-squared-error performance in Figure 5.2 suggests that positional information may be useful in un-supervised sentiment

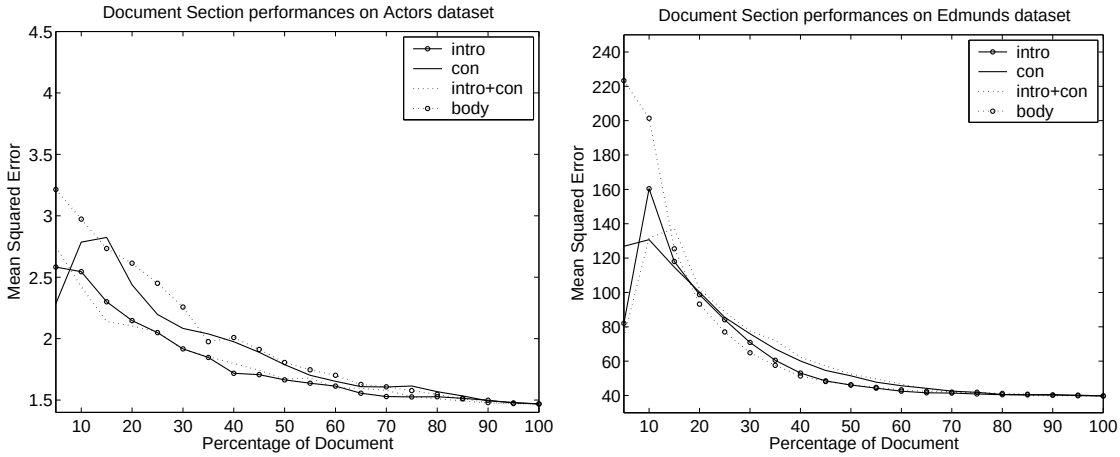

Figure 5.2: Performances of Four Main Sections of a Document

classification of documents. It is, however, important to note that despite potential benefits in exploiting this information, not all corpora possess the same distribution of sentiment rich words. This was clearly depicted in Figure 5.1 where the Actors and Edmunds datasets actually had distributions that contradicted each other. Nonetheless, the results in Figure 5.2 show a strong correlation between supervised sentiment classification performance and the previously charted distributions in Figure 5.1. This therefore opens the possibility that positional information could be useful in an un-supervised sentiment classification context, just so long as one is aware of the underlying distribution of sentiment rich words in the corpus of interest.

5.2 Weighting Phrases by Position

In order to weight a phrase by position, both its location and its documents' length need to be recorded. A weighting function w can then be defined to calculate an appropriate weight for the phrase. Following previous sentiment analysis results by Beineke et al. [2004] where introductions and conclusions were presumed to be the best locations for sentiment rich words, two different implementations of w were instantiated. The first implementation, w_α, gives preference to phrases that occur in the introductions and conclusions of a document, whereas the second one, w_β, prefers those found within the main body. The boundaries of the three sections (introduction, conclusion, and body) were determined by dividing the document into four equal quarters. The first and last quarters were then respectively taken as the introduction and conclusion, whereas the middle half was taken as the body. These proportions were assigned based on the heuristic that the body is

generally the most lengthy part of a document.

In order to describe the two functions w_α and w_β, let $loc_d(b_i)$ be a function that returns the position of the phrase b_i within the test document d. Also let $|d|$ represent the length of document d, and F be an integer such that $F \geq 1$. The first weighting function w_α was then implemented as follows:

$$w_\alpha(b_i, d) = \begin{cases} 1, & \text{if } loc_d(b_i) \leq 0.25|d| \\ 1, & \text{if } loc_d(b_i) \geq 0.75|d| \\ 1/F, & \text{otherwise} \end{cases} \tag{5.1}$$

Note here that the first and last quarters are weighted F times as much as the middle half, i.e. 1 against $1/F$. In contrast, the second function w_β was weighted inverse to this as follows:

$$w_\beta(b_i, d) = \begin{cases} 1/F, & \text{if } loc_d(b_i) \leq 0.25|d| \\ 1/F, & \text{if } loc_d(b_i) \geq 0.75|d| \\ 1, & \text{otherwise} \end{cases} \tag{5.2}$$

Note here that middle half is now weighted F times as much as the first and last quarters, i.e. 1 against $1/F$. With the weighting functions in place, the computation of document orientation in Equation 4.7 of the previous chapter can then be revised as follows:

$$DO(d) = sign\left[\sum_i w(b_i, d) PO(b_i, \mathcal{P}, \mathcal{N})\right] \tag{5.3}$$

Where w can be chosen to be either w_α or w_β.

5.3 Discussion and Summary

The hypothesis behind this chapter is that the location of a phrase within its parent document has an influence on its sentiment strength. The preliminary experiments conducted in this chapter suggest that this hypothesis may be true. They, however, also showed that the probability of finding a sentiment rich word at a specified location varies from domain to domain. For instance, it is more likely to find a sentiment rich word in the first half and last quarter of a document from the Actor dataset. The situation is, however, reversed in documents from the Edmunds dataset, where the body is the most likely source.

The chapter utilises the results from the preliminary experiments to present two weighting function. The first function assigns a higher weight to phrases occurring in the introduction and conclusion, whereas the second function gives priority to phrases occurring in the body.

The evaluation of the weighting functions is presented in chapter 8 and involves only one main experiment. The experiment analyses the effect of the two weighting functions on the unsupervised sentiment classification performance on the Actors and Edmunds datasets.

Chapter 6

A Comparative Study of OSC

Learning to predict either discrete classes or real values from training data has long been an important research topic in Machine Learning [Mitchell, 1997]. In between these two problems, lies a third kind of problem that involves prediction into ordinal classes, i.e. discrete classes with a linear ordering. Surprisingly enough, not much attention has been given to ordinal problems, which is rather disappointing given that many classification problems in the real world fall into this category. A good example of such a problem is Ordinal Sentiment Classification (OSC) which involves predicting into classes that are ordered with respect to sentiment, i.e. ranging from a positive to a negative sentiment orientation.

OSC problems, as with ordinal ones in general, have also been relatively untended in the SAoT community. This owes partly to the fact that until recently, most researchers viewed OSC as being limited to only three classes, i.e. positive, negative, and neutral [Pang et al., 2002]. Furthermore, most studies tacitly assumed that classifiers developed for the more common bi-polar problem would be sufficient to predict documents from the neutral class as these would lie on the boundary. However, the study by Koppel and Schler [2006] disapproves this assumption in that they found classifiers designed for bi-polar problems to be lacking on OSC. Through such efforts, it is now widely accepted that OSC problems can have more than just three classes and that special classifiers are required to address their needs.

It is possible to view an OSC problem as a multi-class classification task, whereby classifiers are built for the purpose of predicting into three or more classes. Note, however, that in order to identify a good classifier for OSC problems, it is first necessary to understand that the distinguish-

ing feature about them is that their classes exhibit similarities that vary with the distance between themselves. For instance, a textual review accompanied by a rating of 1 (on a 10 point scale) is expected to be more similar to one rated at 2 than another at 10. This means that a candidate classifier should focus on achieving two main goals:

1. Maximising the correct predictions.

2. Minimising the distances between actual and predicted classes.

The literature on multi-class classification reveals a rich variety of classification techniques that could be applied to OSC. Amongst these are classifiers that can inherently solve multi-class problems (e.g. Naïve Bayes), and those that decompose the multi-class task into a collection of binary problems, thereafter combining the binary results to formulate a multi-class prediction (e.g. One-vs-All and All-vs-All). Note, however, that although a few isolated studies on OSC have adopted one or more of these techniques [e.g. Koppel and Schler, 2006; Pang and Lee, 2005], there are still no comparative studies that illustrate the superiority of one classifier over the other. This makes it difficult for anyone to make an informed guess regarding which classifiers to employ on an OSC problem.

The main objective of this chapter is to perform a comparative study of the performance of various multi-class classification techniques on OSC problems. The aim is to provide results that can be used as a guide in selecting an appropriate classifier for an OSC problem. In addition to this, the Binary Search Approach (BSA), a multi-class classification technique that is purpose designed for OSC problems is also introduced. The results indicate that the BSA performs as well as other standard multi-class classification techniques. It, however, has a relatively low time complexity which places it amongst the fastest classifiers of its kind.

This chapter proceeds as follows. The next Section discusses the design of the various classifiers that were employed. In this section, both classifiers that decompose a multi-class problem and those that can inherently solve them are discussed. The chapter is then concluded with a summary of these classifiers in Section 6.2. Note that the evaluations are discussed in the Evaluation chapter 8.

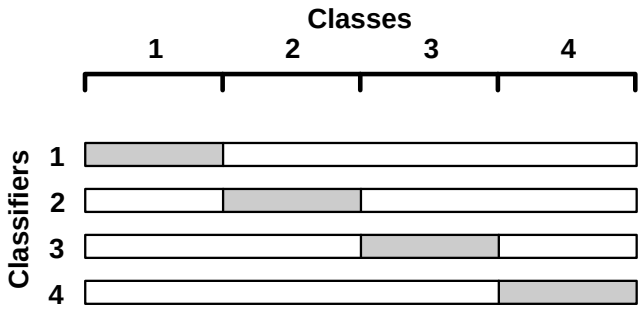

Figure 6.1: The One-vs-All Classifier

6.1 Multi-Class Classifiers for OSC Problems

A standard multi-class classification problem involves a training set consisting of documents belonging to m mutually exclusive classes. The goal is then to construct a classifier which, given a new unseen test document, will correctly predict the class to which the new document belongs. Most text classifiers are binary in nature, and hence can only handle the case where $m = 2$, i.e. binary classification. However, when $m > 2$ then a typical approach is to break down the problem into a series of binary classification tasks whose decisions are then combined to form single multi-class decision.

6.1.1 The One-vs-All Scheme

Amongst the most simple, and well know multi-class classification schemes is the One-vs-All, or the OvA, scheme. Given an m class problem, the OvA scheme would build m real-valued binary classifiers, each one to distinguish the examples in a single class from those in all remaining classes. An unseen document is then classified by passing it to these m classifiers and assigning it the label of the classifier that returns the largest value [Nilson, 1965]. This strategy has been depicted in Figure 6.1 for a 4 class problem. The grey and white regions respectively demarcate the positive and negative classes that each binary classifier is trained on.

Most studies typically use the OvA scheme as a baseline against which to illustrate the superiority of their own algorithms [e.g. Fürnkranz, 2002; Frank and Hall, 2001]. One of its main weaknesses is that its binary classifiers are trained on classes that are inherently skewed in favour of the negative class. Note, however, it is also possible to argue that the OvA scheme isolates each class from the rest hence enabling it to build accurate models of each class. This would in

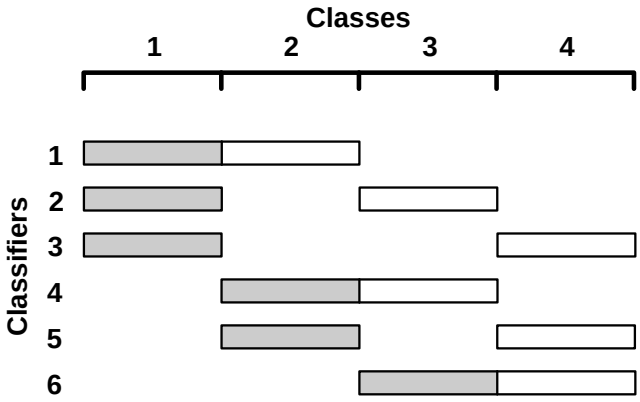

Figure 6.2: The All-vs-All Classifier

turn increase its ability to make *correct predictions*, which is one of the goals in solving an OSC problem.

The One-vs-All scheme both trains and classifies in linear time with respect to the number of input classes. This can been shown by noting that it builds m classifiers for an m-class problem, and hence trains in $O(m)$ time. It then classifies an unseen example by querying all m classifiers, and hence also classifies in $O(m)$ time.

6.1.2 The All-vs-All Scheme

The All-vs-All (AvA) scheme is equally as common as the OvA scheme. It basically converts an m-class problem into a series of binary problems by learning one real-valued binary classifier for each pair of classes, using only training examples from these two classes and ignoring the rest. An unseen document is then classified by sending it to each of the $m(m-1)/2$ classifiers and combining their predictions by simple voting. Figure 6.2 illustrates this strategy for a 4 class problem, where the grey and white demarcations respectively represent the positive and negative classes that each binary classifier is trained on.

The AvA approach has clearly defined semantics, namely to predict whether a test case is more likely to be of class x or class y. Unlike the OvA approach, its binary classifiers are generally free from class skew and hence its performance is relatively unaffected by the number of classes in a given problem [see Fürnkranz, 2002]. The AvA approach considers all pairs of classes separately. This quality enables it to attain the two desired goals for solving an OSC problem: Firstly, it allows the AvA scheme to build accurate models of each class, which improves its ability to make *correct*

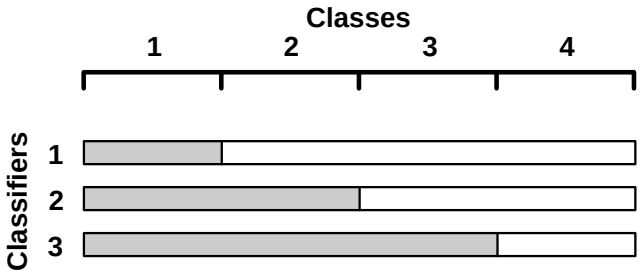

Figure 6.3: The Simple Ordinal Approach

predictions. Secondly, it enables the AvA scheme to preserve inter-class similarities, which in turn aids it in minimising the differences between the *actual and predicted classes*.

The AvA approach both trains and classifies in quadratic time, i.e. $O(m^2)$. This can been shown by noting that it builds $m(m-1)/2$ classifiers for an m-class problem, and hence trains in $O(m^2)$ time. It then classifies an unseen example by querying each of the $m(m-1)/2$ classifiers, and hence also classifies in $O(m^2)$ time. Time complexity is an important factor in choosing between the OvA and AvA techniques. Fürnkranz [2002] claimed that although AvA has a higher time complexity, each of its classifiers have a reduced number of training examples, and hence it works out to be just as fast as the OvA scheme. This is, however, highly dependent the mode of implementation.

6.1.3 The Simple Ordinal Approach

The Simple Ordinal Approach (SOA) takes advantage of the ordinal class structure by dividing the training data on the basis of the linear class ordering [Frank and Hall, 2001]. Given a problem with a set of m ordinal classes $\{c_1, \ldots, c_m\}$, the SOA divides the training data into $m-1$ pairs by incrementally splitting it along the ordinal scale. One real-valued binary classifier is then trained on each split, resulting in $m-1$ classifiers. This idea is illustrated in Figure 6.3 for a four class problem, where the grey and white demarcations respectively represent the positive and negative classes that each binary classifier is trained on.

To classify an unseen document d, all $m-1$ classifiers are queried. The prediction of the j^{th} classifier is then normalised to the range $[0,1]$ and interpreted as the probability $\theta_{d,j}$ of document d occurring in a class subsequent to c_j (i.e. the negative class). The probability that document d belongs to class c_j is then determined by the location of c_j in the linear ordering. There are three

main possibilities of where c_j could occur, and these have been illustrated as follows:

$$P(c_j|d) = \begin{cases} 1 - \theta_{d,j}, & \text{if } j = 1 \\ \theta_{d,j-1} - \theta_{d,j}, & \text{if } 1 < j < m \\ 1 - \theta_{d,j-1}, & \text{if } j = m \end{cases} \tag{6.1}$$

Document d is then labelled with the class that yields the highest probability in Equation 6.1 as follows:

$$label(d) = \underset{c_j}{\operatorname{argmax}} \left[P(c_j|d) \right], \quad j = 1, \ldots, m. \tag{6.2}$$

Note that the binary classifiers of the SOA are mostly trained on skewed data, except for when the number of classes m is even, during which the middle classifier is balanced. Nonetheless, the level of skew varies with each classifier, and decreases to almost zero towards the central classifiers. This feature was intentionally designed in order to narrow down on a test cases' true class by exploiting the linear ordering. For instance, if a given test case x belongs to class 1 in Figure 6.3, then classifier 1 would be most likely to predict it as positive, followed by classifiers 2 and 3 in that order. This consequently enables the SOA to minimise the distances between the actual and predicted classes, which is one of the goals in solving an OSC problem. In contrast to this, classifiers 2, 3, and 4 of the OvA approach in Figure 6.1 would all be equally likely to predict x as positive. This inherently provides the SOA with an advantage over the OvA approach when dealing with OSC problems.

The SOA trains and classifies in linear time with respect to the number of input classes. This can been shown by noting that it builds $m - 1$ classifiers for an m-class problem, and hence trains in $O(m)$ time. It then classifies an unseen example by querying each of the $m - 1$ classifiers, and hence also classifies in $O(m)$ time.

6.1.4 The Binary Search Approach

As with the Simple Ordinal Approach, the Binary Search Approach (BSA) also takes advantage of the ordinal class structure by dividing the training data on the basis of the linear class ordering. This SAoT classification technique was first developed in this research. Given a problem with a set of m ordinal classes $\{c_1, \ldots, c_m\}$, the BSA would recursively divide the training data into two

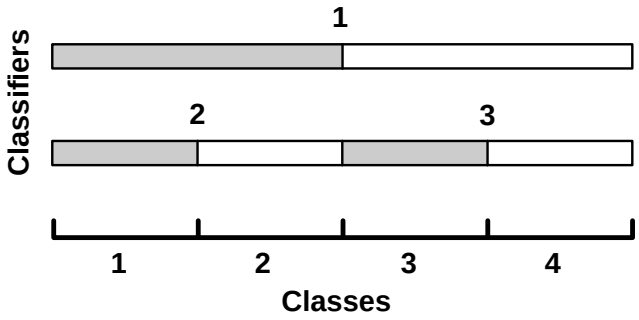

Figure 6.4: The Binary Search Approach

equally sized halves (positive and negative) while respecting the class ordering at each recursion level. One real-valued binary classifier is then trained on each split, resulting in $m - 1$ classifiers. The general idea is illustrated in Figure 6.4 for a four class problem, where the grey and white demarcations respectively represent the positive and negative classes that each binary classifier is trained on.

This divide and conquer strategy can be viewed as a binary tree with $m - 1$ nodes. Assuming this view, a test document would be classified by descending it down from the root node to one of the leaf nodes. The direction of each descent is determined by the output of the classifier at each respective node when queried with the test document. If this output is positive, then the test document would descend to the right, otherwise it would descend to the left. Finally, the classifier output at a leaf node is taken as the class of the test document.

Note that when the number of classes m is not a power of 2, then it may not be possible to always split the training data into two equally sized halves. For instance, if a problem has three ordinal classes $\{c_1, c_2, c_3\}$, then a decision would have to be made to include the middle odd class c_2 into either the positive or the negative side of the split. Two techniques were employed in this regard. In the first technique, the middle odd class was randomly assigned to either the positive or the negative side. This technique was referred to as BSA-RN. In the second technique, the Kullback-Leibler (or KL) distance [see Cover and Thomas, 1991] between the middle odd class and its two neighbouring classes was measured. The middle odd class was then assigned to the side with the neighbour that had the shortest KL-distance to it. This technique was referred to as BSA-KL.

The KL-distance is a measure of the difference between two probability distributions. For

instance, given two distributions P and Q, the KL-distance of Q from P would be defined as:

$$D(P||Q) = \sum_k P(k) \log \frac{P(k)}{Q(k)} \tag{6.3}$$

Using this principle, the middle odd class would be merged with the side of the split whose distribution is closest to its own.

The BSA progressively narrows down on the class of a given test case in a manner that would minimise the ordinal distance between its prediction and the test cases true class. For instance, if a given test case x belongs to class 1 in Figure 6.4, then classifier 1 at the root node would most likely predict it as positive. This reduces the possible error in the final prediction from three units to only one, which is essentially one of the goals of solving an OSC problem. Note, however, that a classification error in the higher nodes is more critical. For instance, if classifier 1 predicts x as negative, then the final error would be at least two units in size. Nonetheless, the BSA is designed such that the upper classifiers are quite general and hence tend to be highly accurate. This reduces the chances of such errors occurring. In addition to this, the BSA hardly experiences any class skew. In the worst case, the difference in size between the positive and negative classes is at most one, and this only occurs when the number of classes m is not a power of 2. This therefore enables the BSA to build accurate class models, hence improving its ability to make correct predictions (another goal of solving an OSC problem).

The BSA trains in linear time and classifies in logarithmic time with respect to the number of input classes. This can been shown by noting that it builds $m - 1$ classifiers for an m-class problem, and hence trains in $O(m)$ time. It then classifies an unseen example by querying at most $\lceil \log m \rceil$ of its classifiers, and hence classifies in $O(\log m)$ time. This makes it amongst the fastest multi-class decomposition techniques employed here.

6.1.5 The Single Classifiers

A single classifier is essentially one that is naturally capable of handling multi-class classification problems. In this research, two single classifiers were used: Naïve Bayes and Regression based Support Vector Machines (SVM-reg).

Naïve Bayes is a simple but powerful classifier that is based on the Bayes Theorem. Given an m-class problem, the Naïve Bayes classifier would build m separate Bayes models, one for

each class. An unseen test document would then be classified by polling the m classifiers and assigning it to the class whose model yields the largest probability. The fact that Naïve Bayes builds separate models for each class allows it to be both accurate, and appreciative of the ordinal inter-class similarities that are present within OSC problems. These two points can be explained by noting that the models that it builds are separate, and also posses the same linear ordering as that of their respective underlying classes. This enables Naïve Bayes to either predict a test cases correct class, or minimise the error of an incorrect prediction. Note that these are the two goals of solving an OSC problem.

SVM-reg approaches OSC from a regression perspective. Given an OSC training dataset, SVM-reg builds a function f that tries to map the training documents to their respective labels as closely as possible. Once this function is learnt, an unseen test document x would then be assigned to the class $c \in \mathcal{C}$ that is closest to the regression value $f(x) \in \mathbb{R}$. Note that this regression value is rounded off because, although OSC can be approached from a regression perspective, it is still a classification problem. As with Naïve Bayes, SVM-reg is also appreciative of the ordinal class structure in that it tries to map the ordinal relationships between the classes of an OSC problem.

Finally, the time complexity details for SVM-reg are beyond the scope of this study. Note, however, that Naïve Bayes trains and classifies in linear time and the details of this can be found in chapter 3.

6.2 Discussion and Summary

The objective of this study is to provide empirical results that can used be in selecting an appropriate classifier for an OSC problem. Two different types of classifiers are explored: Those that decompose a multi-class classification task into a series of binary problems, and those that are inherently capable of handling a multi-class task without having to decompose the classes. The chapter also identifies that a suitable classifier for OSC would have to be capable of maximising its correct predictions, while minimising the error between the actual and predicted classes. A total of six different classifiers were investigated at altogether.

The OvA scheme builds a binary classifier to separate each class from the rest. This fact enables it to build good models of each class, which in turn helps in maximising its correct predictions. Note, however, that each of its models is greatly affected by class skew, especially if

the problem has a large number of classes. This weakens the ability of the OvA approach to minimise the errors in its incorrect predictions. The OvA scheme is relatively fast in that it trains and classifies in linear time.

The AvA scheme builds a binary classifier for each pair of classes. This allows it to respect the linear ordering between the classes, which in turn helps it in minimising the error in its incorrect predictions. In addition to this, the models are based on pairs of classes and hence are not affected by class skew. Note, however, that the AvA scheme is amongst the slowest multi-class classification techniques in this study in that it trains and classifies in quadratic time.

The SOA takes advantage of the class ordering by incrementally splitting the data along the ordinal scale. Thus given an m class problem, the SOA would build $m - 1$ binary classifiers, each trained on one split. Note that the level of skew in the SOA varies with each classifier, and decreases to almost zero towards the central classifiers. This feature enables the SOA to narrow down on a test cases' true class by exploiting the linear ordering. The SOA approach is relatively fast in that it trains and classifies in linear time.

The BSA techniques successively split the training data into two equal halves resulting in binary tree. In the event that two equal halves cannot be obtained, BSA-RN randomly assigns the middle class to one half, whereas BSA-KL uses KL-distance to make the same decision. The divide-and-conquer strategy that the BSA techniques employ enables them to respect the linear class ordering, and hence effectively minimize the errors in their incorrect predictions. They are also quite fast in that they train and classify in linear and logarithmic time respectively.

Naïve Bayes and SVM-reg are the two single classifiers that were employed. Naïve Bayes models each class separately and hence respects the linear ordering. The same is also true with SVM-reg which tries to model the linear ordering between the classes. The time complexity for SVM-reg is beyond the scope of this study. Naïve Bayes, however, trains and classifies in linear time.

The evaluation of the classifiers is presented in chapter 8 and involves two main experiments. The first experiment analyses their MSE performance on the Actors, Edmunds, and Scale datasets. The second experiment analyses their classification speed on the same datasets.

Chapter 7

Feature Selection for OSC Problems

One main characteristic of the Text Classification domain is the problem of high dimensionality. The original feature space in any dataset can easily comprise of several thousand of features. Such high dimensions are impractical for most text classifiers which can only handle a small fraction of the original feature set. For example, the distance computations made by the k Nearest Neighbours classifier, quickly become intractable as the number of features increase. Similar problems are also encountered by other machine learners such as Naïve Bayes, Decision Trees, and Neural Networks, just to name a few. Nonetheless, the problem of feature selection has been well studied by the text classification community [see Yang and Pedersen, 1997; Forman, 2003; Wiratunga et al., 2004]. There now exists several techniques that can effectively reduce the original feature space without sacrificing on classification performance.

Note, however, that despite the significant gains made in the field of feature selection, there are still some text classification problems that still pose a challenge to the field. Such problems normally stem from the fact that most feature selection techniques are based on the assumption that all pairs of classes are equally similar. This is not always the case in the real world, and a good example of a problem that does not adhere to this assumption is the task of Ordinal Sentiment Classification (OSC). Now, being ordinal in nature, OSC problems exhibit inter-class similarities that decay with the ordinal distance between classes. For instance, a textual movie review rated as positive bears a closer resemblance to one rated as medium, than another as negative. This essentially means that the similarity between any pair of classes is a function of the ordinal distance between them. Consequently, by not exploiting this information, most feature selection techniques

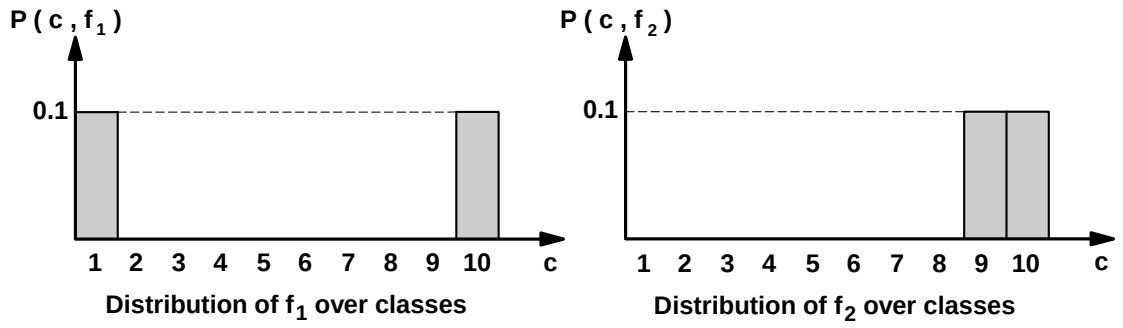

Figure 7.1: Feature f_1 and f_2 posses Distributions that are Equally Important

are inherently incapable of realising their full potential on OSC problems.

The thesis of this chapter is that OSC requires specialised feature selection. The basic argument is that important features in an OSC problem are those that distribute themselves across similar classes. Standard feature selection techniques cannot easily detect such features principally due to their assumption that all classes are equally similar. Under this assumption, a feature that occurs in two similar classes, would be considered as being equivalent to another that occurs in two dissimilar ones, and hence the most appropriate features are overlooked.

The solution that is proposed in this thesis tackles the problem by capitalising on the fact that the ordinal class labels are directly related to the ordinal inter-class similarities. With this knowledge, the probability weight of a given feature in a given class is shared amongst neighbouring classes in manner that decreases with inter-class distance. This smoothing like technique is referred to as the Ordinal Smoothing Procedure (OSP). The collective effect of performing the OSP over all classes is that the distribution of the feature in question peaks if the feature is a member of similar classes, and flattens if it is a member of dissimilar ones. The end result is that, a distribution with a peak would have a higher information content than one that is flat [Shannon, 2001]. This therefore enables standard feature selection techniques to prefer features that occur in similar classes over those that do not.

This chapter proceeds as follows. The next Section discusses the problem of feature selection for OSC into more detail, giving specific examples of where it fails. It then goes on to discuss how the OSP solves the problem. A final summary of the OSP is then presented in Section 6.2. Note that the evaluation of the OSP is discussed in the Evaluation chapter 8.

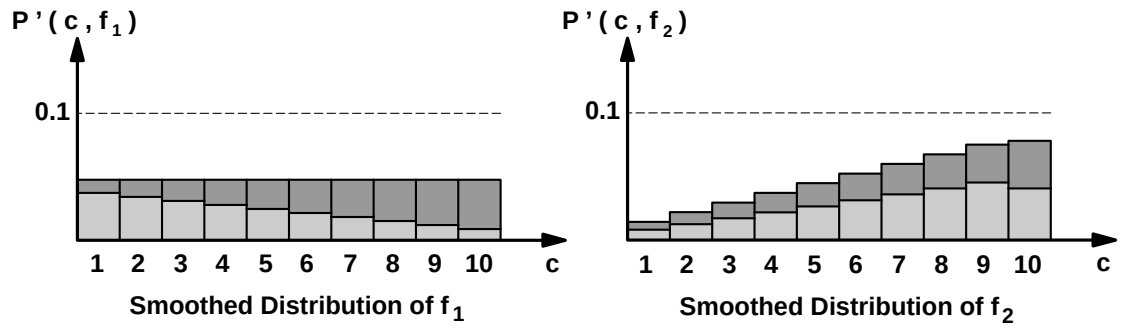

Figure 7.2: Feature f_1 and f_2 are Distinguished by Ordinal Smoothing

7.1 Inter-Class Similarities

In an OSC problem, inter-class similarities, as reflected from the class labels, play an important role in feature selection. To illustrate this using an example, assume that one has two features, f_1 and f_2, that are distributed across a set of equally sized ordinal classes in the proportions shown in Figure 7.1.

If one were to disregard class ordering, and hence inter-class similarities, then these two features would have the same information content, and hence seem equally important. However, with the inclusion of inter-class similarities, then they are different. Clearly, feature f_2 is descriptive of two similar classes and hence is more important than feature f_1 which occurs in two relatively less similar classes. The complication of performing feature selection with algorithms like IG and the CHI score is that they do not exploit inter-class similarities and hence would equate the importance of the two features described in Figure 7.1.

7.1.1 A Solution by Modifying Information Content

The Ordinal Smoothing Procedure solves this problem by smoothing the distributions of the features prior to performing feature selection. This procedure is guided by ordinal inter-class similarities which are inferred from the class labels. It basically disperses the probability weight $P(c_j, f_t)$, of a given feature f_t in a given class c_j, to neighbouring classes in proportions that decay with inter-class similarity. The end result is a distribution that peaks at class c_j and smoothes off as it gets further away. The collective effect of performing the OSP for all classes is that the resultant distribution of feature f_t will tend to be "pyramid shaped" if f_t is descriptive of similar classes, and "flat" if it is descriptive of dissimilar ones. An example of this is given in Figure 7.2

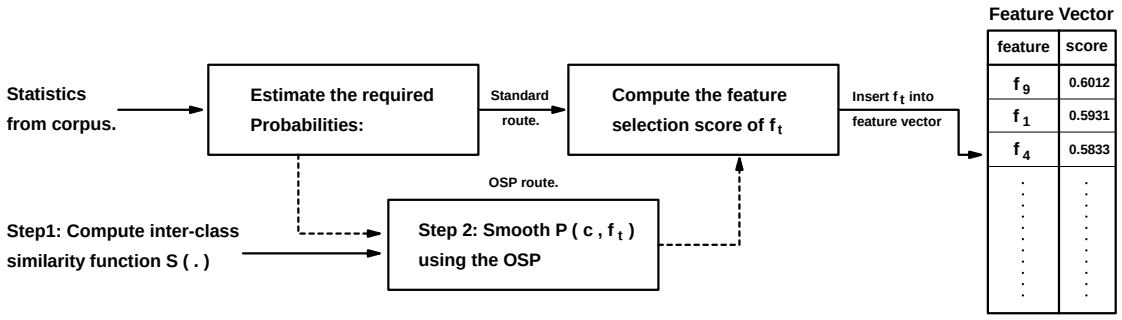

Figure 7.3: The OSP as Applied to a given Feature f_t

which illustrates the outcome of performing the OSP on the distributions of features f_1 and f_2 as introduced in Figure 7.1.

The two shades of colour in this Figure depict the manner in which the probability weights were distributed amongst the classes. Note, for feature f_2, that the cumulative effect of having two peaks at two similar classes is a steep distribution with a peak that spans both classes. This is in contrast to feature f_1 whereby the combined effect of having two peaks in two distant classes is a relatively flat distribution. The overall effect is that the resultant distribution $P'(c, f_2)$ contains a higher information content than $P'(c, f_1)$. This enables both IG and the CHI score to distinguish between the two features.

7.1.2 The Ordinal Smoothing Procedure

Figure 7.3 gives a general overview of the Ordinal Smoothing Procedure as applied to an arbitrary feature f_t. As illustrated in Figure 7.3, the standard feature selection procedure is to estimate the required probabilities and then compute the score for feature f_t. Note, however, that the OSP deviates slightly from this, in that it smoothes the distribution of feature f_t across the classes, $P(c, f_t)$, prior to computing the feature score. There are two steps involved in the OSP:

Step 1

The first step is performed prior to feature selection. It involves initializing the similarity function $S(c_i, c_j)$ with the similarity values between all pairs of classes $c_i, c_j \in \mathcal{C}$, where $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of ordinal classes whose subscripts denote class ordering. Note that the function $S(c_i, c_j)$ acts as a kernel that can be defined in a number of different ways. Ideally, it should model the

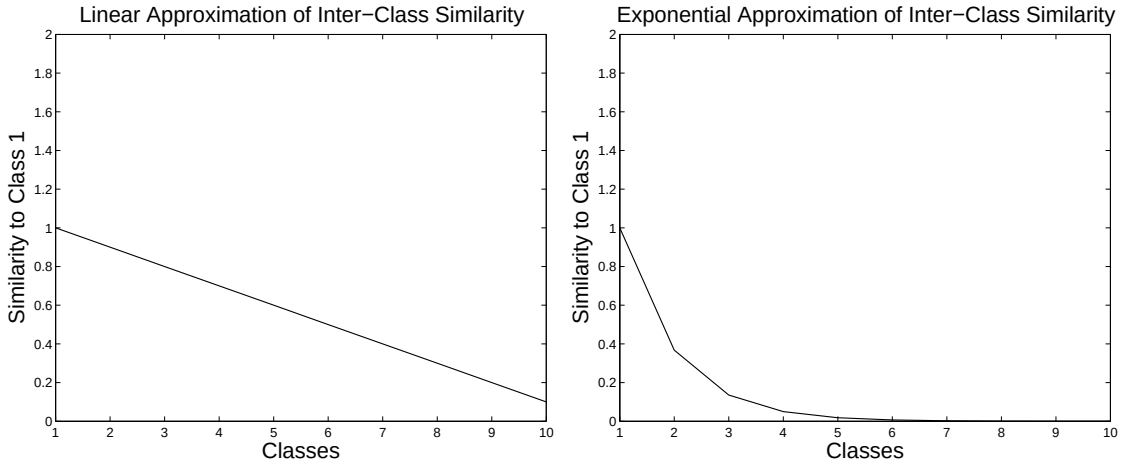

Figure 7.4: Linear and Exponential Approximations of Inter-class Similarity

inter-class similarities of an OSC problem as closely as possible.

Most OSC problems are characterised by inter-class similarities that decay with the distance between classes. These similarities represent the transition of sentiment orientation from one class to the next along the ordinal scale. However, while the existence of this gradual shift in sentiment across the classes is tacitly accepted, the quantification of its actual nature is still, at least for the time being, an unsolved problem in NLP. Nonetheless, due to the need to estimate the underlying inter-class similarities, one could opt for an alternative approximation.

In this research, the similarity function $S(c_i, c_j)$ was implemented in two different ways. The first implementation, S_l, assumes that the similarity between classes c_i and c_j decays linearly with the ordinal distance between them, i.e.

$$S_l(c_i, c_j) = 1 - \frac{|i - j|}{m} \tag{7.1}$$

The second implementation, S_e, assumes that the similarity between classes c_i and c_j decays exponentially, i.e.

$$S_e(c_i, c_j) = \exp\left(-|i - j|\right) \tag{7.2}$$

The similarity curves for both functions have been illustrated in Figure 7.4 for class c_1 against all other classes. Note that the linear function has a more gradual slope that extends across all classes. This has effect of distributing the weight of a feature in a given class to all classes. It, however, also means that the features weight in its class of origin would be much smaller. This

could be disadvantageous if the feature only occurs in one class. It is, however, quite useful in flattening the distributions of features that occur in distant classes. The exponential function has a much sharper slope that rapidly drops to zero. This has the effect of confining the distribution of a features weight in a given class only to close neighbours. This has the advantage of sharpening, or peaking, the distribution of a feature that occurs in only a few neighbouring classes. It is, however, relatively poor at flattening the distribution of a feature that occurs in distant classes.

Step 2

The second step of the OSP is to smooth the distribution of feature f_t across the classes, $P(c, f_t)$, so as to yield a new distribution $P'(c, f_t)$. This is performed at each class $c_j \in \mathcal{C}$ as follows:

$$P'(c_j, f_t) = \sum_k P(c_k, f_t) S(c_k, c_j) \tag{7.3}$$

Where the function S can either assume a value of S_l or S_e. The effect of applying Equation 7.3 to all classes $c \in \mathcal{C}$ for all classes is that the resultant probability weights of feature f_t at each class are stacked against each other with varying proportions of the original weight. If f_t occurs in similar classes, then these classes would inherit relatively large proportions of the original weight which would thus boost the overall information content of feature f_t. However, if f_t occurs in dissimilar classes, then the intermediate classes would receive inversely proportional weights from either of the two dissimilar classes. This would flatten the overall distribution of feature f_t resulting in a lower information content.

Note that the cumulative effect of Equation 7.3 over all classes can be simplified into a single matrix operation as follows:

$$\mathbf{P'} = \mathbf{SP} = \begin{bmatrix} S_{11} & \dots & S_{1m} \\ \vdots & \ddots & \vdots \\ S_{m1} & \dots & S_{mm} \end{bmatrix} \begin{bmatrix} P_1 \\ \vdots \\ P_m \end{bmatrix} \tag{7.4}$$

Here $\mathbf{P}$, $\mathbf{P'}$, and $\mathbf{S}$ respectively denote the original distribution, the smoothed distribution, and the similarity function. Note also that S_{ij} is used as a shorthand for $S(c_i, c_j)$, and P_j for $P(c_j, f_t)$. A useful constraint is to normalize each column of $\mathbf{S}$ so as to ensure that the area under $\mathbf{P'}$ equates to that under $\mathbf{P}$.

7.2 Discussion and Summary

This chapter identifies a new problem with the standard approach to feature selection for OSC problems. It illustrates that important features in OSC problems distribute themselves across similar classes, and then goes on to show that standard feature selection techniques are not designed to identify such features. The reason it gives for this is that inter-class similarities are typically not incorporated into the feature selection procedure. This therefore limits the capabality of distinguishing between important features, that occur in similar classes, and less important ones, that occur in dissimilar classes.

As a solution to this problem, the chapter then proposes the OSP. The OSP is a novel feature selection methodology that has the potential to be coupled with any feature selection technique that makes use of the joint distribution between a feature and the classes. The OSP captures inter-class similarities, implicit in the ordinal scale, into a feature distribution hence enabling standard algorithms to correctly rate the importance of a feature with respect to OSC problems.

It has previously been argued by reviewers that the OSP is simply a revised form of smoothing, which has widely been applied in text classification to tackle issues such data sparseness [see McCallum and Nigam, 1998; Vilar et al., 2004]. Note, however, that the application of smoothing in this context differs from this type of usage in at least two ways. Firstly, smoothing is used here as a pre-bag-of-words procedure in that it is used to establish the words that actually from the bag-of-words. In contrast to this, smoothing in data sparseness is used to avoid zero counts. Assuming that w is a random word and c, a random class, the typical use of smoothing would be to estimate $P(w|c)$ whenever w is unseen in class c (meaning that $P(w|c) = 0$). Secondly, rather that using smoothing to replace unseen word counts, the objective here is to use it as a means of distinguishing between words that are descriptive of similar classes from those that are not. It turns out that, in some cases, making this distinction requires for the OSP to increase word counts in classes where it is already maximal!

The evaluation of the OSP is presented in chapter 8 and involves one experiment which compares the MSE performance of three feature sets. The first feature set (Base) was generated from unmodified feature distributions. The second feature set was from OSP modified distributions using a linear (S_l) similarity kernel, and the last feature set was generated using an exponential (S_e) kernel.

Chapter 8

Evaluation

This chapter reports on the evaluation of the methodologies that were discussed in chapters 4, 5, 6, and 7. Although these evaluations were all performed separately, they can still be roughly classed into two main groups. Chapters 4 and 5 focus on the extraction of sentiment rich phrases, whereas chapters 6, and 7 are based on Ordinal Sentiment Classification. The results obtained within the two respective groups often influenced the subsequent chapter. For instance, the optimal parameters that were found in chapter 4 where used in chapter 5, and similarly those found in chapter 6 were used in chapter 7.

The methodology of these evaluations was discussed in great detail in chapter 3, and the following is a brief summary of the main points that were raised. The overall research employed six real world text datasets: Polarity, Actors, Edmunds, Scale, Reuters, and the Trec Blog Collection. WordNet was also used as a reference dataset for obtaining lexical information. Pre-processing of the datasets included tokenisation as a compulsory step, PoS Tagging and stemming were only performed when specified, and finally conversion to a consisted case and stop-word filtering were always performed. Feature selection was done using Information Gain, the Chi-Squared score, and Document Frequency. The default approach was to use IG to select the top 2% of the feature set. Naïve Bayes and Support Vector Machines were the two supervised classifiers that were used. They were mainly employed to solve OSC problems. The performance of the classifiers was measured using either Mean Squared Error or Accuracy and significance was tested using the two-tailed t-test, at the $p < 0.05$ level. The samples for the significance tests were obtained using 20-folds of cross validation.

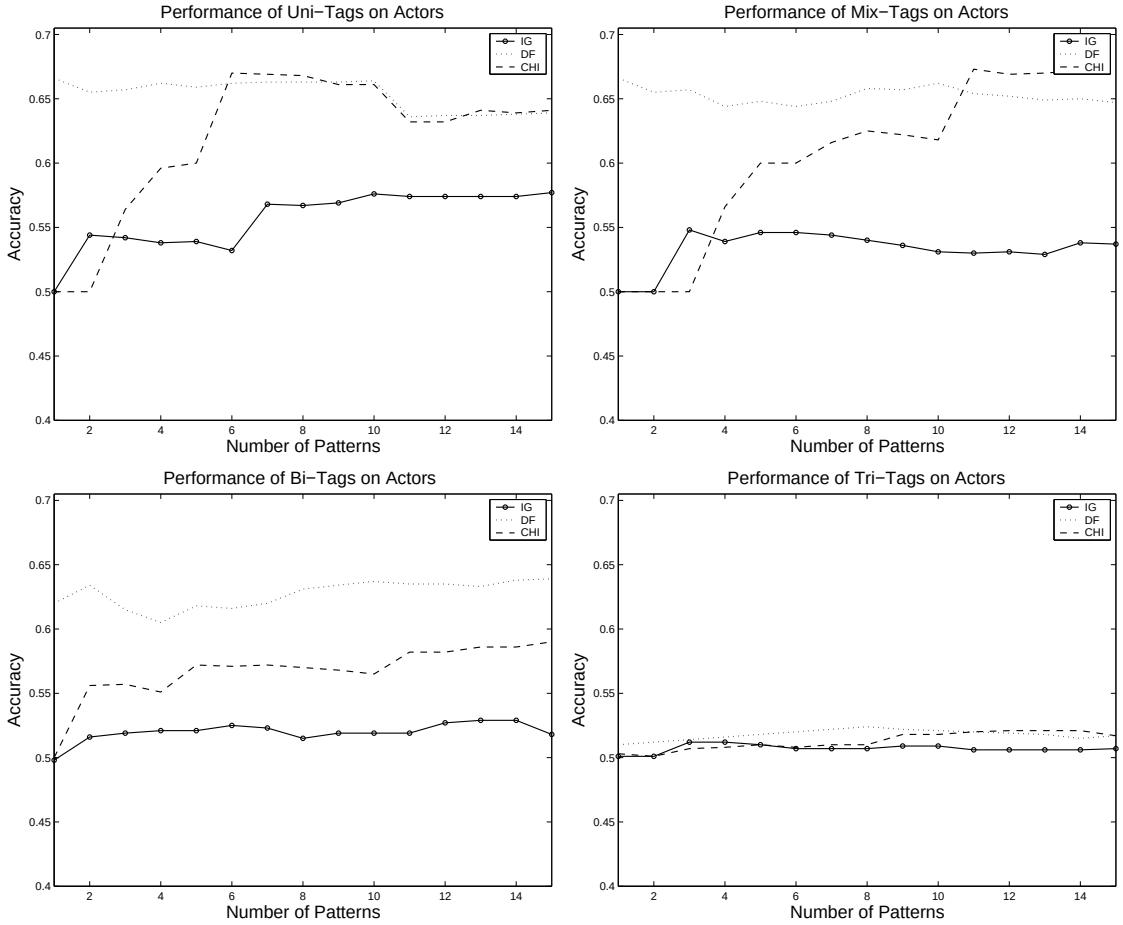

Figure 8.1: Comparison of the Pattern Selection Techniques on Actors

8.1 Evaluation of the PPS Algorithm

Three main experiments were employed to evaluate the pattern selection procedures discussed in chapter 4. The first experiment was a direct comparison between the three automated pattern selection techniques, i.e. IG, CHI, and DF. The second experiment compared the performance manual patterns against that of automatically generated ones. The last experiment compared the effect of a sentiment rich background dataset against that of a non-sentiment rich one.

All three experiments employed bi-polar versions of the Actors and the Edmunds datasets as the un-labelled test examples. These bi-polar versions were derived as follows: For the Actors dataset, classes 1 and 5 were used as the negative and positive classes respectively. For the Edmunds dataset, the negative class was formed from classes 1 and 2 and the positive one was formed from classes 27 and 28. The experiments also used the Trec-Blog Collection as a query

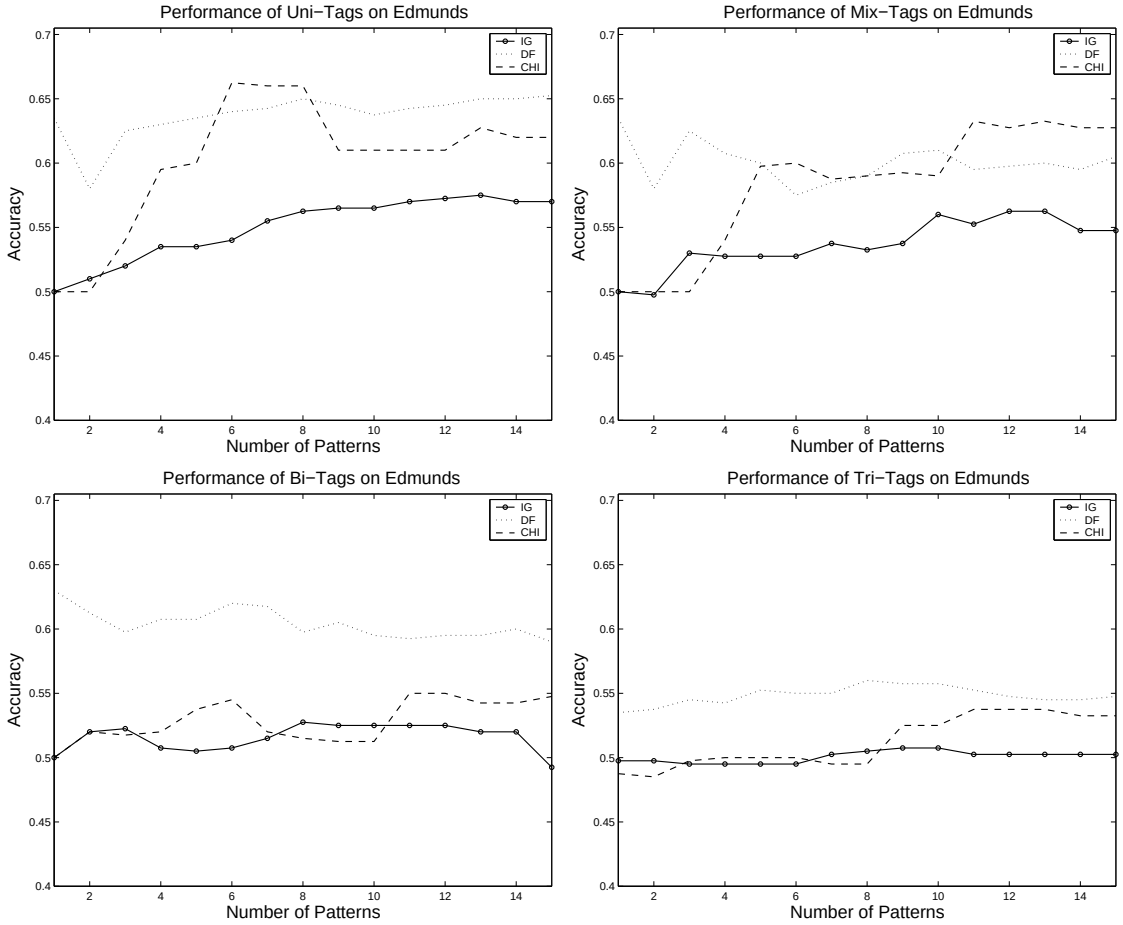

Figure 8.2: Comparison of the Pattern Selection Techniques on Edmunds

dataset in order to obtain the probability estimates required to compute the orientation of a phrase as discussed in Equation 4.2.

8.1.1 Experiment 1: Comparison of Pattern Selection

Figures 8.1 and 8.2 respectively illustrate the average 20-fold classification accuracies achieved while selecting patterns using IG, CHI, and DF. The patterns were selected from the Polarity dataset, and then used to extract phrases from either the Actors or the Edmunds dataset. The Polarity dataset was chosen as it has a large feature set and thus would yield a larger variety of patterns. Also note that the patterns consisted of uni-tags (single PoS tags), bi-tags (two consecutive PoS tags), tri-tags (three consecutive PoS tags), and mix-tags (a mixture of all three).

The results show in both datasets that DF is mostly better than CHI, which in turn is also mostly better than IG. This contradicts several previous studies in feature selection that have reported

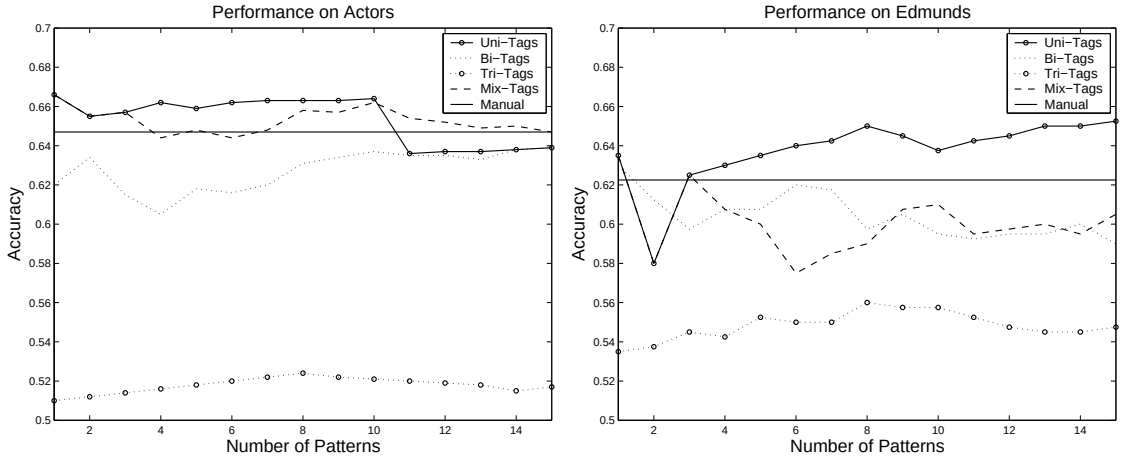

Figure 8.3: Manual vs. Automated Pattern Selection

the opposite trend in performance [Yang and Pedersen, 1997; Sebastiani, 2002; Forman, 2003]. Both IG and CHI are known to return features that are relatively more discriminative than those returned by DF [Forman, 2003]. These results strongly suggest that the discriminative ability of a PoS pattern does not directly influence that of the bi-grams it extracts.

The results also show that uni-tags and mix-tags tend to have the best performance levels, followed by bi-tags and finally tri-tags. Note that this goes against the argument by Turney [2002] which states that a phrase would provide more context than a single word. Nonetheless, it is possible that some information could be missed out by phrases. For instance, the word "Absolutely" has a positive orientation of 1.02. However, given the extract "Do you think it will work? Absolutely" a bi-tag can only extract the word "Absolutely" within the phrase "? Absolutely" which has a lower orientation of 0.79. It is therefore preferable not to omit uni-tags all-together, and this probably explains the good performance of mix-tags. In addition to this, the number of phrases that match bi-tags and tri-tags are expected to be fewer than those that match uni-tags and mix-tags.

8.1.2 Experiment 2: Manual vs. Automated Pattern Selection

Figure 8.3 illustrates the average 20-fold classification accuracies achieved while using either manually or automatically selected PoS patterns. The manual patterns were adopted from the study by Turney [2002] and are listed in Table 4.1, whereas the automated ones were selected using the DF of uni-tags, bi-tags, tri-tags, and mix-tags. Note that the performance of the manual patterns is independent of the x-axis and hence is a straight line. As with the previous experiment, the

No of Patterns	DF			IG			CHI		
	SR	NSR	Diff	SR	NSR	Diff	SR	NSR	Diff
1	0.666	0.586	**+0.080**	0.500	0.515	-0.015	0.500	0.500	0.000
2	0.655	0.655	0.000	0.544	0.548	-0.004	0.500	0.559	**-0.059**
3	0.657	0.657	0.000	0.542	0.548	-0.006	0.564	0.559	+0.005
4	0.662	0.657	+0.005	0.538	0.537	+0.001	0.596	0.568	+0.028
5	0.659	0.662	-0.003	0.539	0.536	+0.003	0.600	0.614	-0.014
6	0.662	0.662	0.000	0.532	0.536	-0.004	0.670	0.610	**+0.060**
7	0.663	0.661	+0.002	0.568	0.565	+0.003	0.669	0.609	**+0.060**
8	0.663	0.659	+0.004	0.567	0.567	0.000	0.668	0.616	**+0.052**
9	0.663	0.664	-0.001	0.569	0.567	+0.002	0.661	0.594	**+0.067**
10	0.664	0.667	-0.003	0.576	0.570	+0.006	0.661	0.594	**+0.067**
11	0.636	0.664	**-0.028**	0.574	0.577	-0.003	0.632	0.596	**+0.036**
12	0.637	0.664	**-0.027**	0.574	0.578	-0.004	0.632	0.596	**+0.036**
13	0.637	0.664	**-0.027**	0.574	0.579	-0.005	0.641	0.602	**+0.039**
14	0.638	0.636	+0.002	0.574	0.580	-0.006	0.639	0.600	**+0.039**
15	0.639	0.635	+0.004	0.577	0.579	-0.002	0.641	0.594	**+0.047**

Table 8.1: The Performance of Sentiment Rich Background Data on Actors (uni-tags)

Polarity dataset was also employed as the sentiment rich background corpus.

The results show that uni-tags are, on average, slightly better than manual patterns on both datasets whereas mix-tags are only better on the Actors dataset. This therefore places uni-tags as having the average best performance, followed by manual patterns, mix-tags, bi-tags, and finally tri-tags. The high performance of manual patterns is not unexpected as experience shows that manual pattern construction is both rigorous and time-consuming [see Jackson and Moulinier, 2002;Section 4.2]. Each pattern, once derived, must be tested against a representative collection and fine tuned in light of the results. This is an iterative process that must be done by a domain expert. It is therefore not surprising to expect a better performance when using such carefully designed patterns. However, the advantage of the automated patterns is that they reduce the demand on the knowledge engineer. This makes it suitable in applications whereby the data structure morphs rapidly making it infeasible to employ hand-crafted techniques.

8.1.3 Experiment 3: Effects of a Sentiment Rich Background

Tables 8.1, 8.2, 8.3 and 8.4 illustrate the classification accuracies achieved on Actors and Edmunds while using either a sentiment rich (SR) or a non-sentiment rich (NSR) background dataset. The Polarity dataset was used as the SR background, whereas Reuters was used as the NSR back-

No of	DF			IG			CHI		
Patterns	SR	NSR	Diff	SR	NSR	Diff	SR	NSR	Diff
1	0.635	0.520	**+0.115**	0.500	0.508	-0.008	0.500	0.510	-0.010
2	0.580	0.580	0.000	0.510	0.562	-0.052	0.500	0.502	-0.002
3	0.625	0.625	0.000	0.520	0.562	-0.042	0.540	0.502	+0.038
4	0.630	0.610	+0.020	0.535	0.540	-0.005	0.595	0.525	**+0.070**
5	0.635	0.620	+0.015	0.535	0.562	-0.028	0.600	0.528	**+0.072**
6	0.640	0.620	+0.020	0.540	0.562	-0.022	0.662	0.542	**+0.120**
7	0.642	0.618	**+0.025**	0.555	0.598	-0.042	0.660	0.542	**+0.118**
8	0.650	0.620	**+0.030**	0.562	0.578	-0.015	0.660	0.542	**+0.118**
9	0.645	0.632	+0.012	0.565	0.575	-0.010	0.610	0.542	**+0.068**
10	0.638	0.632	+0.005	0.565	0.582	-0.018	0.610	0.542	**+0.068**
11	0.642	0.638	+0.005	0.570	0.575	-0.005	0.610	0.552	**+0.058**
12	0.645	0.638	+0.008	0.572	0.580	-0.008	0.610	0.552	**+0.058**
13	0.650	0.638	+0.012	0.575	0.580	-0.005	0.628	0.545	**+0.082**
14	0.650	0.642	+0.008	0.570	0.588	-0.018	0.620	0.542	**+0.078**
15	0.652	0.645	+0.008	0.570	0.582	-0.012	0.620	0.558	**+0.062**

Table 8.2: The Performance of Sentiment Rich Background Data on Edmunds (uni-tags)

ground. Each cell in the SR and NSR columns represent the average of 20 folds of cross validation. The difference between these two columns is shown in the Diff column with all statistically significant differences[1] shown in bold. Uni-tags were used in the first two tables whereas Mix-tags were used in the last two. These were chosen as they were found to be the most effective in the previous two experiments.

Note, in both test datasets, that both DF and CHI are on average more effective when using a sentiment rich background dataset. This is true for both uni-tags and mix-tags. The effect is more pronounced with CHI, which performs significantly better on almost all pattern sizes when using a SR background dataset. IG on the other hand does not perform very well with SR data. However, its performance is the lowest of the three pattern selection techniques making it a less attractive option. Nonetheless, the general trend is that DF and CHI have the highest performances and also that they perform better on a SR background dataset. This offers strong evidence of the importance of using a SR background dataset.

[1]Measured using the two tailed t-test at the $p < 0.05$ level.

No of Patterns	DF			IG			CHI		
	SR	NSR	Diff	SR	NSR	Diff	SR	NSR	Diff
1	0.666	0.586	**+0.080**	0.500	0.501	-0.001	0.500	0.500	0.000
2	0.655	0.655	0.000	0.500	0.502	-0.002	0.500	0.498	+0.002
3	0.657	0.645	+0.012	0.548	0.496	**+0.052**	0.500	0.497	+0.003
4	0.644	0.645	-0.001	0.539	0.495	**+0.044**	0.566	0.497	**+0.069**
5	0.648	0.639	+0.009	0.546	0.496	**+0.050**	0.600	0.496	**+0.104**
6	0.644	0.639	+0.005	0.546	0.506	**+0.040**	0.600	0.495	**+0.105**
7	0.648	0.639	+0.009	0.544	0.548	-0.004	0.616	0.497	**+0.119**
8	0.658	0.645	+0.013	0.540	0.548	-0.008	0.625	0.497	**+0.128**
9	0.657	0.636	+0.021	0.536	0.545	-0.009	0.622	0.498	**+0.124**
10	0.662	0.631	**+0.031**	0.531	0.548	-0.017	0.618	0.498	**+0.120**
11	0.654	0.625	+0.029	0.530	0.556	-0.026	0.673	0.498	**+0.175**
12	0.652	0.635	+0.017	0.531	0.556	-0.025	0.669	0.504	**+0.165**
13	0.649	0.635	+0.014	0.529	0.594	**-0.065**	0.670	0.503	**+0.167**
14	0.650	0.635	+0.015	0.538	0.594	**-0.056**	0.672	0.504	**+0.168**
15	0.647	0.635	+0.012	0.537	0.594	**-0.057**	0.669	0.504	**+0.165**

Table 8.3: The Performance of Sentiment Rich Background Data on Actors (mix-tags)

8.2 Evaluation of Positional Information

The two positional weighting schemes discussed in chapter 5 were evaluated by comparing their unsupervised sentiment classification performance on bi-polar versions of the Actors and Edmunds datasets. Following from the results in the previous section, only uni-tags selected from the Polarity dataset using DF were used here as these were largely found to be the most effective. Most of the other default experimental settings were the identical to those in the previous section.

Figure 8.4 illustrates the average 20-fold classification accuracies achieved by the two weighting functions for uni-tags of size 5, 11, and 15 (The complete set of results are given in Figure A.3 and A.4 of the General Appendix). The results indicate that performance on the Actors dataset benefits with w_α, but looses with w_β. This clearly corresponds with the distribution in Figure 5.1 of chapter 5 where sentiment rich words within Actors documents are mainly concentrated within the front half with a few spikes in the conclusions. A similar correspondence is also present within the Edmunds dataset which benefits more from the w_β function as most of its sentiment rich words are concentrated in the central body of its documents.

Nonetheless, despite the apparent benefits suggested by the above results, a pre-requisite to weighting a phrase by its locality is knowledge of the distribution of sentiment rich words within documents of the domain of interest. Without this knowledge, performance improvements of

No of Patterns	DF			IG			CHI		
	SR	NSR	Diff	SR	NSR	Diff	SR	NSR	Diff
1	0.635	0.520	**+0.115**	0.500	0.498	+0.002	0.500	0.510	-0.010
2	0.580	0.580	0.000	0.498	0.498	0.000	0.500	0.535	**-0.035**
3	0.625	0.582	+0.042	0.530	0.478	**+0.052**	0.500	0.538	**-0.038**
4	0.608	0.590	+0.018	0.528	0.462	**+0.065**	0.540	0.530	+0.010
5	0.600	0.590	+0.010	0.528	0.470	**+0.058**	0.598	0.532	+0.065
6	0.575	0.598	-0.022	0.528	0.475	**+0.052**	0.600	0.532	+0.068
7	0.585	0.592	-0.008	0.538	0.505	+0.032	0.588	0.535	+0.052
8	0.590	0.602	-0.012	0.532	0.505	+0.028	0.590	0.538	+0.052
9	0.608	0.590	+0.018	0.538	0.492	+0.045	0.592	0.530	**+0.062**
10	0.610	0.580	+0.030	0.560	0.490	**+0.070**	0.590	0.518	**+0.072**
11	0.595	0.582	+0.012	0.552	0.510	+0.042	0.632	0.505	**+0.128**
12	0.598	0.575	+0.022	0.562	0.510	+0.052	0.628	0.495	**+0.132**
13	0.600	0.575	+0.025	0.562	0.522	+0.040	0.632	0.482	**+0.150**
14	0.595	0.585	+0.010	0.548	0.522	+0.025	0.628	0.482	**+0.145**
15	0.605	0.585	+0.020	0.548	0.522	+0.025	0.628	0.502	**+0.125**

Table 8.4: The Performance of Sentiment Rich Background Data on Edmunds (mix-tags)

any weighting function would only be by chance. Consequently, it is important to sample the distribution of the domain, as was done here, prior to employing any weighting function.

8.3 A Comparative Study on OSC

The multi-class classification techniques that were discussed in chapter 6 were compared along two main dimensions: (1) Their MSE performance, and (2) Their classification speed. The comparisons were performed between OvA, AvA, SOA, BSA, Naïve Bayes, and SVM-reg. Note that the underlying binary classifiers of OvA, AvA, SOA, and BSA were built using both Naïve Bayes and SVM-reg leading to a total of 12 multi-class classification schemes altogether. All the experiments were performed on the three standard ordinal datasets, i.e. Actors, Edmunds, and Scale.

8.3.1 Experiment 1: Comparison of MSE Performance

Figure 8.5 illustrates the MSE performances achieved by the 12 multi-class classification techniques when evaluated on the three standard datasets. The initial feature sets were reduced to the indicated sizes using IG. Each point in the graphs represents the average of 20 folds of cross validation.

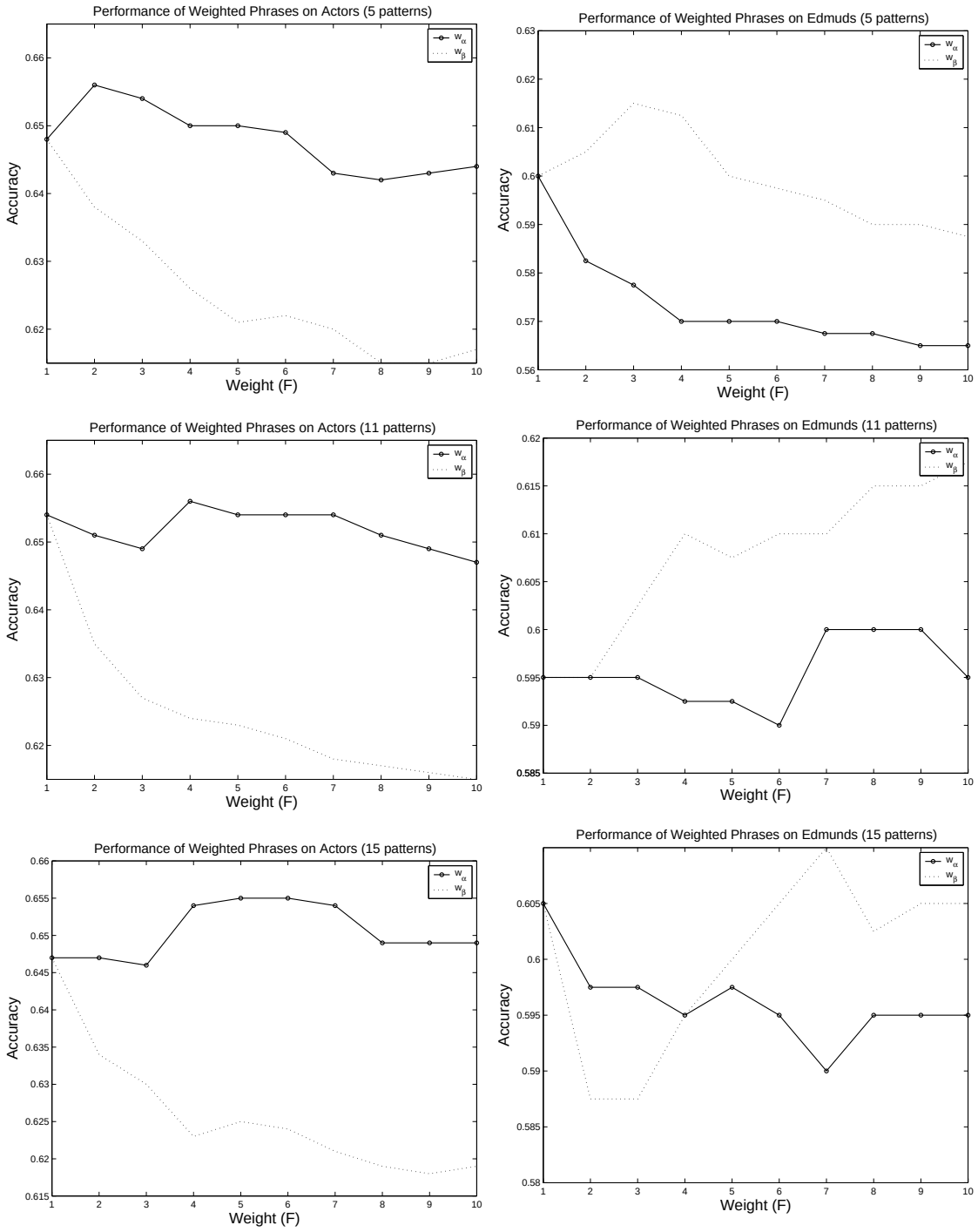

Figure 8.4: Performance of the Weighting Functions on the Datasets

The most distinct feature about the results is that SVM-reg yields the best performances in 2/3 datasets, and is only marginally outperformed by the SOA, implemented using Naïve Bayes, in the Scale dataset. This outstanding performance is not surprising because SVM-reg is well known

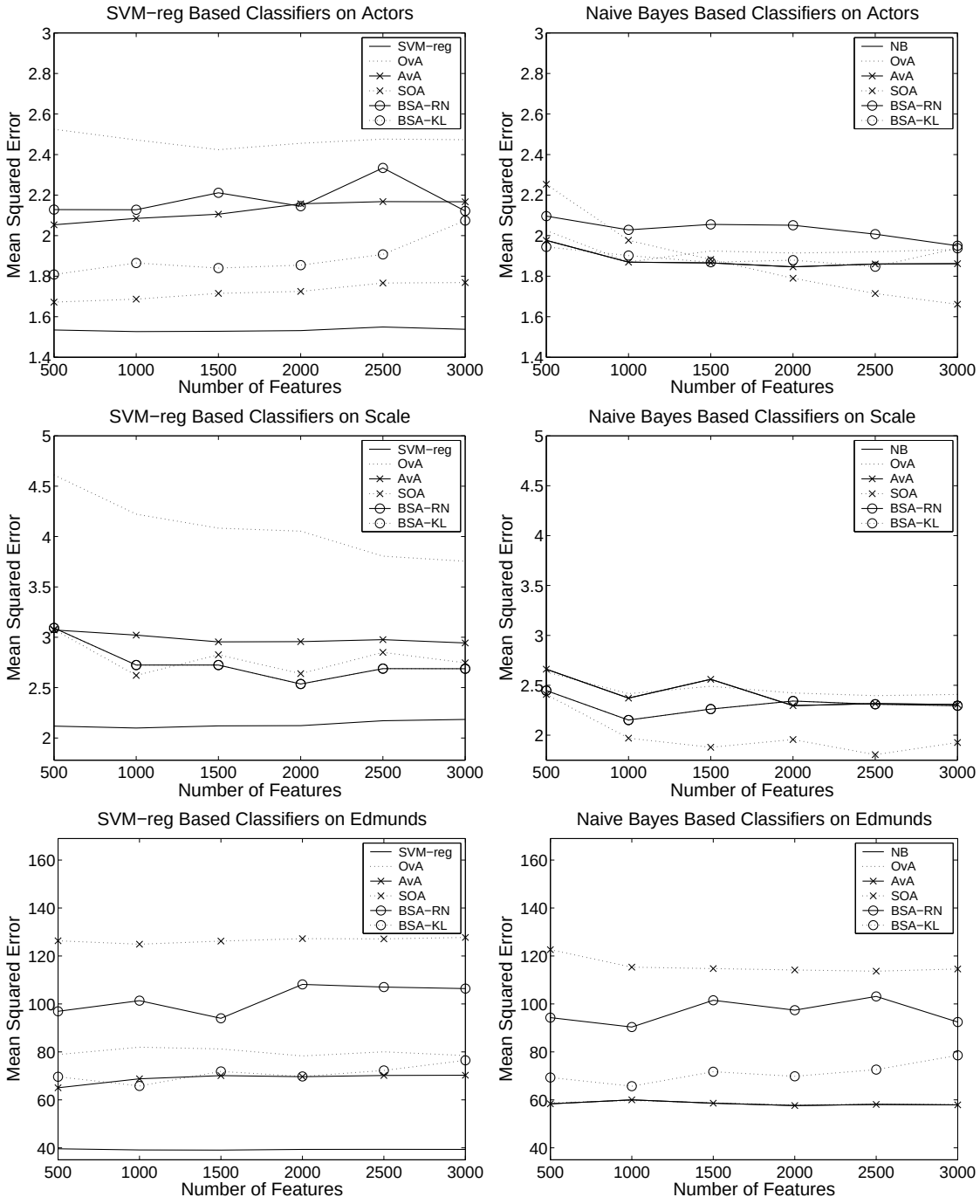

Figure 8.5: MSE of SVM-reg and Naïve Bayes based classifiers on OSC

for its effectiveness and is considered to be the current state-of-the-art in classification.

Another distinct feature about the results is that the SVM-reg based schemes tend to have a more varied performance than the Naïve Bayes based ones, which suggests that Naïve Bayes is more stable than SVM-reg. This result confirms observations made in previous studies whereby

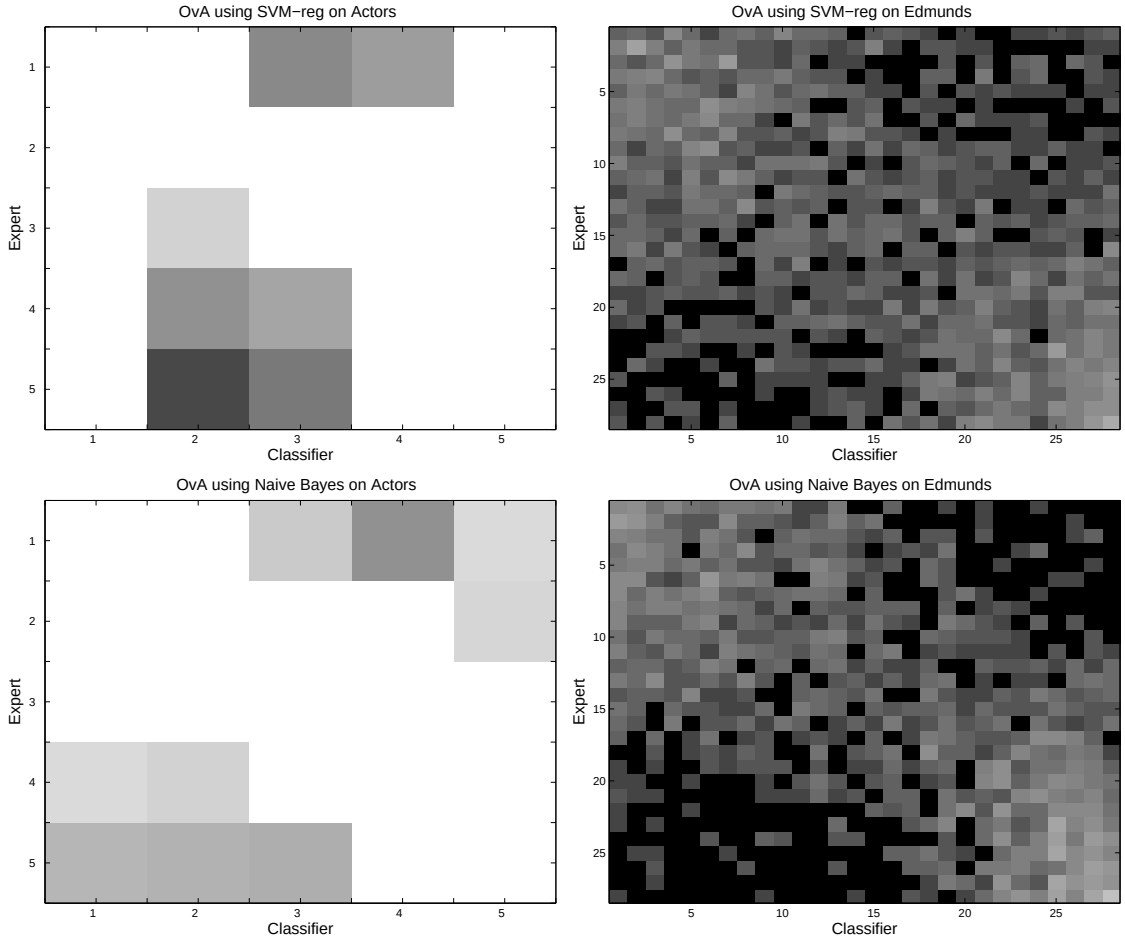

Figure 8.6: Confusion Matrices for OvA on Actors and Edmunds

Naïve Bayes was found to be highly robust when subjected to various test conditions [Mukras, 2004]. It is for this reason that Naïve Bayes is a favourite choice for industrial applications that require reliable classification techniques.

Note that the OvA approach generally had the poorest performance in all datasets except Edmunds. Here, it performed quite well coming in third when implemented using SVM-reg, and first when using Naïve Bayes. This result is slightly unexpected for the reason that the Edmunds dataset has the highest class skew, and hence one would expected the OvA approach to perform poorest here. In order to investigate this result further, the confusion matrices generated over 20-folds of cross validation on the Actors and Edmunds datasets while using 2% of the initial feature set were plotted as shown in Figure 8.6. A confusion matrix compares a classifiers prediction against expert judgements on a class-by-class basis. The entries in its non-diagonal cells are indicative of the classifiers incorrect predictions, and hence the best scenario is when all entries are confined

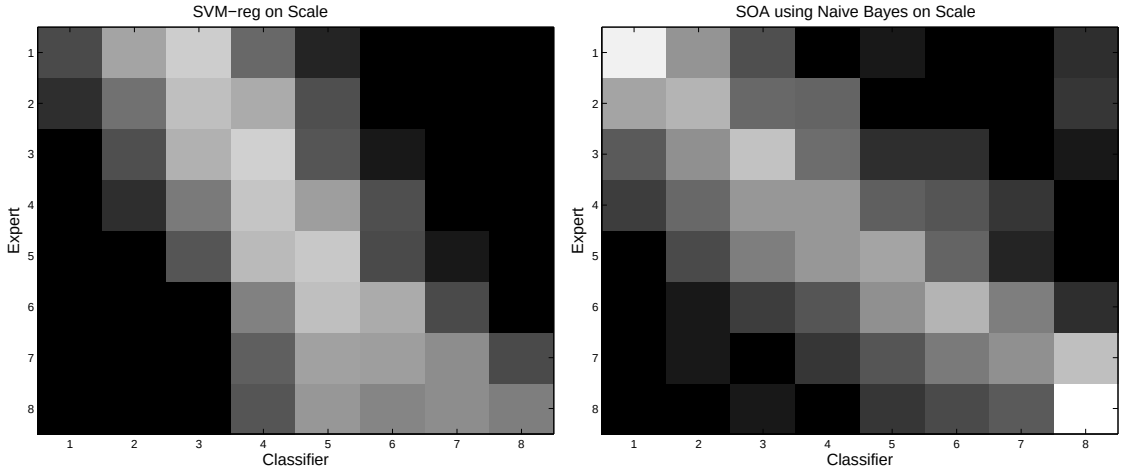

Figure 8.7: Comparison of SVM-reg and SOA on the Scale Dataset

to the diagonal cells. In Figure 8.6, the entries in each cell are represented by a gray-scale colour scheme; the darker the shade, the fewer the entries. Ideally all cells except those on the diagonal should be dark, as this indicates total agreement between the expert and the classifier.

From the Actors confusion matrix, it is clear that the OvA approach has a high preference for the two extreme classes, i.e. positive and negative. This probably highlights the difficulty in classifying neutral documents. In addition to this, the small margin of classes in the Actors dataset means that most documents would most likely resemble those from the extreme classes [Koppel and Schler, 2006]. Note also, in the Edmunds confusion matrix, that there exists a broad band along the main diagonal. Furthermore, this broad band is divided into two main clusters, one at each end of the main diagonal. These two clusters clearly prove that, as with the Actors dataset, the OvA approach also prefers the extreme classes of the Edmunds dataset. However, the large number of classes in this dataset means that the incorrect predictions are distributed over a large area. This therefore reduces the overall MSE, and enables the OvA approach to perform relatively well on the Edmunds dataset.

The performance of AvA does not fluctuate as much as that of OvA, and it seems to be centred around the average across all datasets. It is interesting to note that AvA implemented using Naïve Bayes is identical in performance to Naïve Bayes. This can be explained by noting that AvA retains the original classes. Consequently, the class models that it builds are identical to those of Naïve Bayes and hence their identical performance.

The SOA performs relatively well on most datasets and even outperforms SVM-reg on the

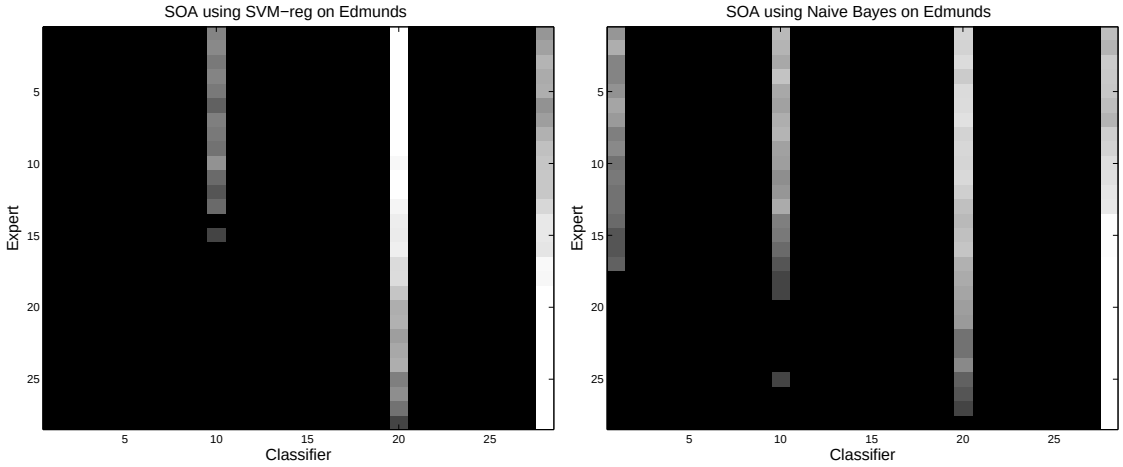

Figure 8.8: Confusion Matrices for SOA on Edmunds

Scale dataset when implemented using Naïve Bayes. In Figure 8.7 one can clearly see that although the SOA confusion matrix has a broader diagonal, it is relatively more well aligned diagonal than that of SVM-reg. Note, however, that the SOA drops in performance on the Edmunds dataset. The confusion matrices in Figure 8.8 reveal that this performance drop is due to its unusual preference for classes 1, 10, 20, and 28. This is probably due to two reasons: Firstly, as was experienced by the OvA approach, most documents tend to resemble to the two extremes polarities and hence the preference of SOA for classes 1 and 28. Secondly, the high number of predictions for classes 10 and 20 clearly mean that most intermediate class documents resemble these two classes to the SOA. One possible explanation for this is that the step-wise approach of the SOA (see Figure 6.3) coupled with the high level of class skew in the Edmunds dataset causes it bundle all intermediate class documents into the classes that lie between the central class and the extreme classes, i.e. classes 10 and 20. It would however be necessary to perform further analysis to ascertain these explanations and this is a possible area for future work.

As for the two BSA techniques, it is clear that using KL-distance to assign the middle odd class (i.e. BSA-KL) is better than simply doing so randomly (i.e. BSA-RN). BSA-KL performed quite well, and its Naïve Bayes implementation was observed to be at par with SVM-reg on the Scale dataset. It is worth noting that, in the Scale dataset, both BSA techniques have the same performance. The reason for this is that the Scale dataset has 8 classes, which is equal to the third power of two, i.e. 2^3. This therefore means that all its binary splits are even is size and hence both BSA-RN and BSA-KL would yield the same binary trees.

Classifier	Dataset	Single	OvA	AvA	SOA	BSA-RN	BSA-KL
SVM-reg	Scale	**260.1**	1714.0	1231.9	1453.2	650.0	669.0
	Actors	**748.1**	7066.4	1640.8	3986.6	1643.0	1736.3
	Edmunds	**816.2**	13598.2	12689.6	25225.5	2795.2	2841.2
NB	Scale	**130.4**	382.8	493.0	351.0	284.0	304.1
	Actors	**136.2**	361.9	360.8	312.4	276.2	294.8
	Edmunds	**155.8**	2015.7	5668.6	1993.2	835.4	871.8

Table 8.5: Time Performance in Milliseconds Per Cross-Validation Fold

8.3.2 Experiment 2: Comparison of Classification Speed

Table 8.5 illustrates the average time in milliseconds that each classifier took to train and test on over each of the 20 folds of cross validation. For each dataset, the initial feature set was reduced by 98% using IG. For each row of each dataset, the performances significantly ($p < 0.05$) better than the rest are shown in bold.

The most distinct aspect of the results is that Naïve Bayes, implemented as a single classifier, attained the highest speed. The results also show that it was at least twice as fast as any other classifier on all respective datasets. In addition to this, the Naïve Bayes based classifiers are also much faster than their respective SVM-reg based counterparts. These results are highly encouraging, and serve to further support the evidence that Naive Bayes is a highly robust classifier that can be as good as other relatively much more complex alternatives.

The second fastest classifiers were the BSA techniques. This was expected as they had the lowest time complexity, i.e. $O(\log m)$. Understandably, BSA-RN was faster than BSA-KL which had to contend with the overhead of computing the KL-distance. Nonetheless, its MSE was much better than that of BSA-RN.

For the SVM-reg based techniques, the BSA schemes were followed by AvA approach. Note, however, AvA was the slowest classifier for the Naïve Bayes based techniques. The reason for this difference is that the speed of the SVM-reg on a given classification task is more dependent on the number of documents, whereas that of Naïve Bayes is more dependent on the number of classes. Hence, since AvA reduces the test set into *several small binary problems*, its SVM-reg based version is quite fast as the problems are quite small. Conversely, it's Naïve Bayes based version is much slower as the classes are numerous in number.

Dataset	Vec size	IG			CHI_1			CHI_{m-1}		
		Base	S_l	S_e	Base	S_l	S_e	Base	S_l	S_e
Edmund	5	55.57	54.99	56.61	59.10	**55.94**	60.83	65.76	64.60	**61.37**
	75	41.92	41.83	41.81	44.49	**42.80**	45.28	64.91	56.75	**44.55**
	150	40.74	39.83	40.22	42.68	**40.46**	42.67	64.83	56.71	**42.57**
	600	39.02	**38.51**	39.09	39.84	**39.23**	39.97	60.57	48.16	**39.89**
	1100	38.84	38.81	38.98	39.05	38.87	39.41	55.46	44.22	**39.40**
	1200	38.86	38.78	38.98	39.32	38.92	39.38	55.45	43.89	**39.37**
	3000	39.30	39.08	39.19	39.10	38.88	39.39	53.31	41.79	**39.38**
Scale	5	5.28	5.53	5.51	5.03	4.92	5.50	5.17	5.35	5.55
	75	2.81	2.89	3.14	3.99	**3.56**	4.24	4.50	**3.92**	4.36
	150	**2.53**	2.65	2.74	3.34	**2.80**	3.73	3.98	3.37	3.54
	600	**2.21**	2.33	2.45	2.54	**2.35**	2.64	2.99	2.62	2.57
	1100	2.24	2.36	2.30	2.32	2.33	2.39	2.83	2.35	2.39
	1200	2.21	2.37	2.32	2.30	2.32	2.36	2.83	2.32	2.33
	3000	2.26	2.29	2.29	2.29	2.29	2.31	2.31	2.29	2.32
Actors	5	1.91	1.92	1.90	1.90	1.90	1.99	**1.89**	1.98	1.99
	75	1.58	1.55	1.57	1.62	**1.58**	1.62	1.65	1.67	1.63
	150	1.51	1.53	1.54	1.56	**1.48**	1.54	1.63	1.55	1.53
	600	1.47	1.48	1.46	1.49	1.53	1.51	1.55	1.51	1.50
	1100	1.45	1.47	1.47	1.51	1.49	1.49	1.50	1.48	1.47
	1200	1.46	1.47	1.47	1.48	1.49	1.49	1.53	1.48	1.47
	3000	1.52	1.49	1.46	1.49	1.46	1.46	1.50	1.48	1.50

Table 8.6: The Impact of the OSP while using SVM-reg

8.4 Evaluation of the OSP

The OSP was evaluated on three OSC datasets, i.e. Actors, Edmunds, and Scale. Its impact was analysed by comparing Naïve Bayes and SVM-reg performance on three feature vectors that differed only in the modifications made to the feature distributions that they were generated from. The first feature vector (Base) was generated from unmodified feature distributions. However, the second and third were generated from OSP modified distributions using either a linear (S_l) or an exponential (S_e) similarity kernel respectively. Note also that three feature selection heuristics were used, i.e. IG, CHI_1, and CHI_{m-1}. This resulted in a total of nine feature vectors per dataset.

Table 8.6 and 8.7 respectively illustrate the results that were obtained when SVM-reg and Naïve Bayes were applied to the three OSC datasets. Each cell, in these two Tables, represents the average MSE that was obtained in 20 folds of cross validation. For each vector size of each feature selection heuristic (IG, CHI_1, or CHI_{m-1}), the performances significantly ($p < 0.05$) better than the rest are shown in bold. The complete set of results from which those in Table 8.6 and 8.7 were

Dataset	Vec size	IG			CHI_1			CHI_{m-1}		
		Base	S_l	S_e	Base	S_l	S_e	Base	S_l	S_e
Edmund	5	200.1	194.8	**152.0**	170.4	**136.5**	147.2	245.3	234.4	**148.2**
	75	74.50	75.94	**67.22**	74.73	**68.45**	73.57	210.3	129.3	**71.66**
	150	64.64	63.90	60.36	67.44	**59.86**	66.85	203.1	127.9	**65.00**
	600	58.61	54.87	53.79	58.30	**53.65**	59.61	171.5	87.83	**59.15**
	1100	57.45	54.98	**51.79**	57.36	**51.89**	57.63	119.6	72.49	**56.45**
	1200	57.41	54.77	**51.55**	58.79	**52.14**	56.25	119.1	70.10	**56.04**
	3000	56.96	55.47	**49.05**	55.81	**52.84**	56.50	105.3	61.96	**55.71**
Scale	5	10.75	11.68	**9.12**	10.29	9.18	11.61	16.92	**9.98**	11.87
	75	4.31	3.82	4.07	5.47	4.98	6.29	10.95	**5.11**	6.32
	150	3.05	3.21	3.57	4.54	**3.45**	5.24	6.35	4.50	4.86
	600	2.62	2.43	2.91	3.35	**2.50**	2.97	4.11	3.17	3.17
	1100	2.88	**2.55**	2.87	2.76	2.63	2.84	3.82	2.71	2.77
	1200	2.84	2.58	2.80	2.81	2.54	2.92	3.74	2.66	2.84
	3000	2.48	2.33	2.37	2.66	**2.25**	2.74	2.75	2.48	2.55
Actors	5	3.75	3.81	3.94	4.15	3.97	4.83	4.17	4.36	4.82
	75	2.44	2.40	2.46	2.54	2.37	2.41	2.67	**2.51**	2.37
	150	2.22	**2.10**	2.22	2.28	2.13	2.13	2.47	2.14	2.13
	600	1.98	1.88	1.88	1.97	1.87	1.85	2.22	1.89	1.89
	1100	1.96	1.81	1.77	1.94	1.81	1.79	2.00	1.84	1.82
	1200	1.95	1.75	1.80	1.92	1.81	1.78	1.97	1.83	1.79
	3000	1.87	1.74	1.81	1.91	1.76	1.74	2.09	1.74	1.81

Table 8.7: The Impact of the OSP while using Naïve Bayes

obtained are respectively illustrated in Figure A.1 and A.2 of the General Appendix.

Between the three feature selection heuristics, the effect of the OSP was, on average, felt most by CHI_{m-1} then CHI_1 and finally IG. Note, however, that an opposite trend was observed with MSE performance in that IG was at the top, followed by CHI_1 and lastly CHI_{m-1}. This performance trend is not surprising as IG has been know to perform better than CHI [Sebastiani, 2002; Yang and Pedersen, 1997]. In addition to this, this trend also helps to explain the relatively larger impact that the OSP had on the two CHI techniques. This was probably because the OSP manifests the information content within the distribution of a feature, and this could have been of benefit the CHI technique. However, being information theoretic in nature, IG already has the baseline capability of utilising information content and hence the OSP would have a lesser impact on it.

As for the three datasets, the impact of the OSP is felt most by Edmunds followed by Scale and finally Actors. This trend can be explained by noting that the Edmunds dataset extends across

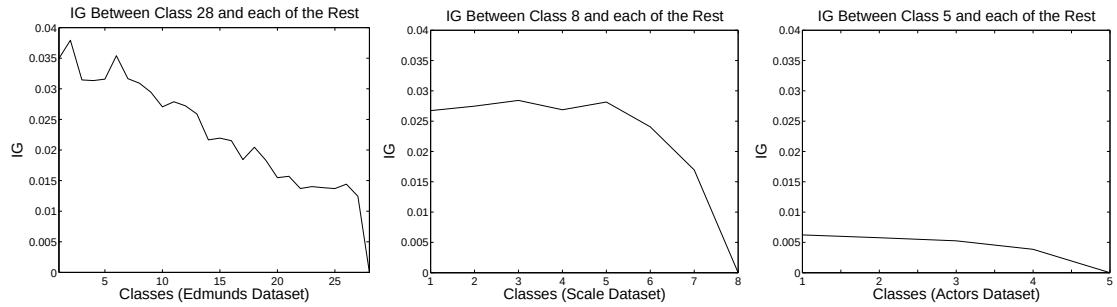

Figure 8.9: IG between the Extreme Positive and all Other Classes

a wider number classes, meaning that its inter-class similarities would also possess an equivalently larger range. This is clearly evident in Figure 8.9 where each graph illustrates the average IG of the top 2% features contained in the most positive class and each of the other classes. Note that the IG of the Edmunds dataset spans about 3.5×10^{-2} units, whereas those of the Scale and Actors datasets respectively cover about 2.5×10^{-2} and 5.0×10^{-3} units. Consequently, one would expect the Edmunds dataset to be relatively more responsive to the OSP as its wide inter-class similarity bounds makes it easier to distinguish between features that occur in nearby classes over those that do not. Conversely, at the other extreme, the small inter-class similarity bounds of the Actors dataset makes its features resemble those of an orthogonal dataset. This makes it difficult to distinguish between features that occur in nearby classes from those that do not. Consequently, the Actors dataset benefits least from the OSP due to this reason.

The overall results suggest that the OSP has a positive impact on the performance of the Base distributions. There are only 3 out of 126 instances in Table 8.6 and 8.7 where the Base distributions were found to be significantly better than the OSP modified ones. This is overwhelmingly in favour of the OSP. Consequently, there is strong evidence in support of the initial hypothesis that important features in an OSC problem distribute themselves across similar classes.

A supplementary experiment was performed with the aim of investigating the individual features that were affected by the OSP. The most outstanding feature that was found was the word "smoothest" from the Edmunds dataset which was promoted by about 2,300 positions up the IG feature ranking by the OSP. A study of the corpus revealed that this feature was highly concentrated within positive reviews occurring in contexts such as "*It is without a doubt the smoothest riding vehicle I have ever driven.*" Hence, due to its uni-polar occurrence, its distribution was revised so to yield an IG score that would be relatively higher than if its presence was divided

between distant classes.

8.5 Summary

This chapter evaluates all the methodologies that were discussed throughout the thesis. The PPS algorithm, which automates the process of PoS pattern selection, was evaluated by a series of 3 experiments. The first of these investigated three pattern selection techniques: IG, CHI, and DF. DF was found to be most effective, followed by CHI, and lastly IG. This experiment also found patterns of either unit length or a mixture of lengths (one, two, or three) to yield the best performances. The second experiment compared PPS generated patterns against manually formed ones and found the two to be roughly equivalent. The last experiment compared the performance of a sentiment rich dataset against that of a non-sentiment one and found the former to be crucial in the process of selecting PoS patterns.

The experiments on the effect of weighting phrases with respect to their locality in a document found it to be useful to do so. However, it was also found to be necessary to first sample the distribution of sentiment rich words in a corpus prior to adopting a weighting criterion.

The single classifiers (SVM-reg and Naïve Bayes) were found to be more effective than the decomposition ones on OSC problems. This was true for experiments involving both MSE and Time performance. The main reason behind this was that the single classifiers were found to be inherently capable respecting inter-class similarities. This therefore meant that decomposing them would probably degrade their natural predictions. There were however a few cases, such as the SOA implemented using Naïve Bayes, where the decomposition classifiers were significantly better.

Amongst the decomposition classifiers, the BSA was found to be average with regard to MSE performance. It was, however, the fastest with regard to time performance. This was mainly due to its superior time complexity, i.e. logarithmic as compared to others which were linear at best.

The experiments on the OSP found it to be capable of improving the performance of both IG and CHI on OSC problems. The two similarity kernels, S_l and S_e, both had a positive effect on the performance of one or more of the feature selection techniques. This success rate has important implications in that it offers the possibility of adopting relatively more advanced kernels that could yield further improvements.

Chapter 9

Conclusions and Future Work

This research has investigated the five objectives that were listed out in the introduction, with the sixth one being the evaluation. In this chapter, these objectives are compared against the work that was eventually achieved during the course of the study. Comments and recommendations are made regarding contributions of the methodologies that have been presented. Finally, some issues that deserve further investigation are discussed.

9.1 The Objectives Revisited

Each of the objectives that were undertaken in this research have yielded at least one contribution to the discipline of Sentiment Analysis of Text. Firstly, two new ordinal datasets have been added to the community of OSC. These have both been employed in internationally published work and have proven to be important test beds for OSC research. Secondly, the PoS pattern selector was developed. Thirdly, the importance of position in sentiment classification of text was established. Fourthly, a ranking of suitable classifiers for OSC was formulated. In addition to this the BSA was also added into the research community. Lastly, the Ordinal Smoothing Procedure, which revolutionises the manner in which feature selection is performed in OSC, is contributed.

In this section, each of the objectives that led to these various contributions are revisited and a summary of how they were addressed is given. In addition to this, recommendations of how the resultant methodologies should be employed.

1. **To compile two ordinal SAoT datasets:** Two ordinal SAoT datasets, i.e. Actors and Ed-

munds, were successfully compiled. These two datasets, along with the Scale dataset developed by Pang and Lee [2005] in a separate research, served as the primary test-beds for all experiments involving OSC. They have also been employed in at least two other publications [see Chakraborti et al., 2007; Mukras et al., 2007].

2. **To research and develop an automated PoS pattern selection algorithm:** The PPS algorithm, which automatically selects PoS patterns, was implemented and evaluated. This algorithm operates by first applying PoS tags to the words of a labelled sentiment-rich-corpus. The words in this corpus are then omitted, after which normal feature selection techniques such as DF, CHI, and IG are used to select PoS patterns of various sizes. The empirical results in Experiment 2 showed that the resultant patterns were at least *as good as* the manually selected ones. This consequently fulfils the current objective.

   The results in Experiment 1 also revealed some interesting findings regarding the effectiveness of the feature selection techniques. DF was found to be, on average, the most effective followed by CHI and then IG. These results contradict previously observed trends in traditional feature selection where IG and CHI have consistently outperformed DF [see Yang and Pedersen, 1997; Forman, 2003]. One can therefore conclude that document frequent PoS patterns are better extractors of sentiment than discriminative ones.

   Experiment 1 also showed that uni-tags and mix-tags tend to perform best, followed by bi-tags and finally tri-tags. This trend is possibly due to the relative sparsity of phrases that match bi-tags and tri-tags as opposed to those that match uni-tags and mix-tags. Lastly, Experiment 3 showed that it was important to use a sentiment rich background corpus as opposed to a non-sentiment rich one. This was shown to be true for both uni-tags and mix-tags.

3. **To investigate possible approaches of weighting the importance of a phrase with respect to its position:** Two techniques of weighting phrases with respect to position were implemented. The first gave preference to phrases occurring in the central body, whereas the second gave preference to those occurring in the front and rear. While the weighting techniques were found to be beneficial to performance, they were also found to be domain specific in that each domain required its own weighting criterion. It is thus necessary to sample the domain prior to adopting a weighting scheme.

Consequently, in as far as illustrating that the weighting of phrases can improve performance, the current objective was met. Note, however, that the research did not yield an all purpose weighting algorithm, but rather presented a heuristic that can be used to do so.

4. **To perform a comparative study of multi-class classification techniques as applied to OSC problems:** Two types of classifiers for OSC problems were evaluated: Those that are naturally capable of handling multi-class problems (single classifiers), and those that decompose the task into multiple of binary problems thereafter combining the resultant decisions into one global decision. The BSA, a new decomposition technique that is purpose designed for OSC, was also amongst those evaluated.

   Experiment 1 of the evaluation indicated that the single classifiers, and in particular SVM-reg, generally had a better MSE performance than the decomposition techniques. Experiment 2 also showed the single classifiers to be much faster as well. Naïve Bayes was typically the fastest with classification times twice as fast as any other classifier. As for the BSA, its MSE performance was generally average. It was, however, the fastest amongst the decomposition techniques due to its logarithmic time complexity, as opposed to the others which were either linear or quadratic.

5. **To implement a novel feature selection technique for OSC:** The follow up study on this objective illustrates that important features in OSC problems distribute themselves across similar classes. The study also shows that standard feature selection techniques would equate the importance of such features against those that are distributed across dissimilar classes.

   The Ordinal Smoothing Procedure (OSP) is then proposed as a solution to this problem. The OSP re-organises the distribution of a feature such that its information content becomes reflective of the classes across which it is distributed. This is done by dispersing the probability weight of each class to all other classes in a manner that decays with ordinal distance. The result is that features that occur in dissimilar classes would obtain flat distributions, whereas those that occur in similar classes would obtain relatively peaked distribution. This then makes it possible for feature selection algorithms such as Information Gain and the Chi-Squared score to distinguish between such features. Experimental results on three real world datasets confirm that important features in OSC problems occur in similar classes,

and that utilising such features can lead to significant performance improvements.

9.2 Future Work

The methods presented in this research constitute a significant contribution to the domain of SAoT. A complete understanding and robust implementation of these methods may assist researchers in developing better SAoT algorithms, and also users in solving difficult real world SAoT problems. Nonetheless, there is still much scope for further development and the following sections highlight a number of issues that have been identified for future work.

9.2.1 Beyond Tag Adjacency

The patterns selected by the PPS algorithm were found to be at least as good as the manually selected ones. Note, however, that only uni-tags and mix-tags were able to significantly outperform the manual patterns, whereas bi-tags and tri-tags were mostly poorer. This is possibly due to the relative sparsity of phrases that match bi-tags and tri-tags in comparison to those that match uni-tags and mix-tags. A possible solution for this could be to relax the consecutive tag constraint within certain linguistic contexts. For instance, given the sentence "I felt confident and safe in their company" the bi-tag `JJ NN` would, after pre-processing, only extract one the phrase, "safe company." However, if the algorithm were to detect the two conjoined adjectives, *confident* and *safe*, and relax the consecutive tag constraint, then the same bi-tag would extract two phrases, "confident company" and "safe company." Such rules could significantly reduce the sparsity of the phrases extracted by bi-tags and possibly tri-tags.

The results illustrated the positive effects of weighting phrases with respect to position. However, the main limitation of this was that the domain had to be sampled prior to adopting a weighting criteria. One way of overcoming this could be to have a collection of document templates with learnt weighting distributions. A new document collection could then be assigned the weighting distribution of the template that is most similar to it. One could possibly even use a Genetic Algorithm to arrive at the optimal weighting distribution. This would, however, require more than just three divisions along the text as was done in this research.

9.2.2 Beyond OSC

The BSA was implemented as two different variants, BSA-KL and BSA-RN. The MSE performance of BSA-KL was found to be consistently better than that of BSA-RN, probably because BSA-KL uses the KL-distance to assign middle odd classes, whereas BSA-RN does so randomly. An interesting issue is whether further improvements in performance could be achieved by employing other distance measures, such as cosine or Euclidean, to assign the middle odd classes. It would also be interesting to see if any improvements can be made by using the majority vote of various distance measures.

The OSP disperses the probability weights of each feature in each class to neighbouring classes in a manner dictated by the heuristic that models the inter-class similarity kernel. In this research, this kernel was implemented to model either linear or exponential inter-class similarities. Note that both these heuristics are synthetic. An interesting alternative experiment would be to work with with the actual, rather than synthetic, interclass similarities. One possibility is to mine the similarities directly from the data by use of some metric such inter-class IG (such as was done in Figure 8.9), or a measure based on the semantic differences within the text contained in the various ordinal classes. These measures would, however, need to be mapped into some surrogate representation to make them useable.

Another possibility of future work looks at extending the OSP to areas other than just ordinal problems. Given that one could mine inter-class similarities from the data, then it should be possible to work various other problems such as those that are hierarchical, or those that constantly morph with time (such as in news filters). So long as one can directly infer the inter-class similarities, then our procedure can, at least in theory, be extended to almost any type of problem. Clearly there is much potential in exploring these avenues, and the results in this thesis suggest promising returns.

References

P. Baxendale. Machine-Made Index for Technical Literature—An Experiment. *IBM Journal*, pages 54–361, 1958.

P. Beineke, T. Hastie, C. Manning, and S. Vaithyanathan. An Exploration of Sentiment Summarization. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications (AAAI Technical Report SS-04-07)*, pages 12–15, 2004.

T. Briscoe and J. Carroll. Robust Accurate Statistical Annotation of General Text. In *Proc. of LREC*, pages 1499–1504, Las Palmas, Canary Islands, May 2002.

S. Chakraborti, R. Mukras, R. Lothian, N. Wiratunga, S. Watt, and D. Harper. Supervised Latent Semantic Indexing using Adaptive Sprinkling. In *Proc. of IJCAI*, pages 1582–1587. AAAI Press, 2007.

K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, 1990. ISSN 0891-2017.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

P. Domingos and M. J. Pazzani. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In *Proc. of ICML*, pages 105–112, 1996.

D. Donlan. Locating Main Ideas in History Textbooks. *Journal of Reading*, 24(2):135–140, 1980.

H. P. Edmundson. New methods in automatic extracting. *Journal of the ACM*, 16(2):264–285, 1969.

C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *JMLR*, 3:1289–1305, 2003.

E. Frank and M. Hall. A Simple Approach to Ordinal Classification. In *Proc. of EMCL*, pages 145–156, London, UK, 2001. Springer-Verlag.

J. Fürnkranz. Pairwise Classification as an Ensemble Technique. In *Proc. of ECML*, pages 97–110, London, UK, 2002. Springer-Verlag.

E. Gabrilovich and S. Markovitch. Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs Competitive with C4.5. In *Proc. of ICML*, pages 321–328, Banff, Alberta, Canada, 2004. Morgan Kaufmann.

V. Hatzivassiloglou and K. McKeown. Towards the Automatic Identification of Adjectival Scales: Clustering Adjectives According to Meaning. In *Proc. of the ACL*, pages 172–182, 1993.

V. Hatzivassiloglou and K. McKeown. A quantitative evaluation of linguistic tests for the automatic prediction of semantic markedness. In *Proc. of the 33rd annual meeting on ACL*, pages 197–204, Morristown, NJ, 1995. ACL.

V. Hatzivassiloglou and K. R. McKeown. Predicting the Semantic Orientation of Adjectives. In *Proc. of EACL*, pages 174–181. ACL, 1997.

V. Hatzivassiloglou and J. M. Wiebe. Effects of Adjective Orientation and Gradability on Sentence Subjectivity. In *Proc. of Computational Linguistics*, pages 299–305, Morristown, NJ, USA, 2000. ACL.

P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. In *Proc. of IAAI*, pages 49–64. AAAI Press, 1991.

D. A. Hull. Stemming algorithms: a case study for detailed evaluation. *J. Am. Soc. Inf. Sci.*, 47 (1):70–84, 1996.

P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization*. John Benjamins, 2002.

T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of ECML*, pages 137–142, London, UK, 1998. Springer-Verlag.

J. S. Justeson and S. M. Katz. Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text. *Natural Language Engineering*, 1:9–27, 1995.

J. Kivinen, M. K. Warmuth, and P. Auer. The Perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. *Artif. Intell.*, 97(1-2):325–343, 1997.

M. Koppel and J. Schler. The Importance of Neutral Examples in Learning Sentiment. *Computational Intelligence*, 22(10):100–109, 2006.

A. Lehrer. Markedness and Antonymy. *Journal of Linguistics*, 31(1):397–421, September 1985.

C. Macdonald and I. Ounis. The TREC Blogs06 Collection : Creating and Analysing a Blog Test Collection. Technical report, Department of Computing Science, University of Glasgow, Glasgow, UK, 2006.

C. Macdonald, I. Ounis, and I. Soboroff. Overview of the TREC-2007 Blog Track. In *Proc. of the TREC*, 2007.

A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Proc. of AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998.

T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

R. Mukras. A Comparison of Machine Learning Techniques Applied to Sentiment Classification. Master's thesis, University of Sussex, Falmer, Brighton, UK, 2004.

R. Mukras, N. Wiratunga, and R. Lothian. Selecting Bi-Tags for Sentiment Analysis of Text. In *Proc. of AI-2007*, pages 181–194, Cambridge, England, 2007a. Springer.

R. Mukras, N. Wiratunga, R. Lothian, S. Chakraborti, and D. Harper. Information Gain Feature Selection for Ordinal Text Classification using Probability Re-distribution. In *Proc. of IJCAI Textlink Workshop*, 2007b.

R. Mukras, N. Wiratunga, and R. Lothian. The Robert Gordon University at the Opinion Retrieval Task of the 2007 Trec Blog Track. In *Proc. of Trec*, 2007.

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

N. Nilson. *Learning Machines*. McGraw Hill, 1965.

J. J. Paijmans. Relative weights of words in documents. In L. G. M. Noordman and W. A. M. de Vroomen, editors, *STINFON*, 1994.

B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proc. of ACL*, pages 271–278, 2004.

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*, pages 115–124, 2005.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proc. of EMNLP*, pages 79–86, 2002.

C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975. ISSN 0001-0782.

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

C. E. Shannon. A Mathematical Theory of Communication. *Mobile Computing and Communications Review*, 5(1):3–55, 2001.

P. D. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proc. of EMCL*, pages 491–502, London, UK, 2001. Springer-Verlag.

P. D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Un-supervised Classification of Reviews. In *Proc. of ACL*, pages 417–424, Morristown, NJ, USA, 2002. ACL.

D. Vilar, H. Ney, A. Juan, and E. Vidal. Effect of Feature Smoothing Methods in Text Classification Tasks. In *Proc. of PRIS*, pages 108–117, Porto, Portugal, 2004.

J. M. Wiebe. Tracking point of view in narrative. *Comput. Linguist.*, 20(2):233–287, 1994.

J. M. Wiebe, R. F. Bruce, and T. P. O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proc. of the 37th annual meeting of the ACL on Comp. Ling.*, pages 246–253, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

T. Wilson, J. Wiebe, and R. Hwa. Just How Mad Are You? Finding Strong and Weak Opinion Clauses. In *Proc. of AAAI*, pages 761–769. AAAI Press, 2004.

N. Wiratunga, I. Koychev, and S. Massie. Feature Selection and Generalization for Retrieval of Textual Cases. In *Proc. of ECCBR*, pages 806–820. Springer-Verlag, 2004.

Y. Yang and J. O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In *Proc. of ICML*, pages 412–420. Morgan Kaufmann, 1997.

Appendix A

General Appendix

A.1 Comparing Performance

In this research, all significance tests were performed using the two tailed t-test. The two paired samples required for this test were typically were obtained using 20 folds of cross validation. This section discusses the manner in which these two concepts can be implemented.

Cross Validation

Cross-validation, sometimes called rotation estimation, is the statistical practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset, while the other subset(s) are retained for subsequent use in confirming and validating the initial analysis. The initial subset of data is called the training set; the other subset(s) are called validation or testing sets. There are several different variants of cross validation and the one used here is known as K-fold cross validation.

In K-fold cross-validation, the original sample is partitioned into K subsamples. Of the K subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $K - 1$ subsamples are used as training data. The cross-validation process is then repeated K times (the folds), with each of the K subsamples used exactly once as the validation data. The K results from the folds then can be averaged (or otherwise combined) to produce a single estimation.

The Two Tailed t-test

Given two sets of data-points, one can use the t-test to determine whether the means are distinct, provided that the underlying distributions can be assumed to be normal. The data-points are normally paired, so that each member of one set has a unique relationship with a particular member of the other set (e.g., the performance of a classifier after two different feature selection heuristics).

The two tailed t-test is conducted by first assuming that the two sets have the same mean. The p-value is then looked up from the computed t-statistic, and if $p < 0.05$, then one can reject the initial assumption and conclude that the means are different. In contrast to this, the upper tailed t-test has an initial assumption that first mean is greater than the second. Consequently, a rejection of this assumption can only lead to the conclusion that the first mean is not larger than the second, with the possibility that they could be statistically the same.

All results in this research were reported using the two tailed t-test at the $p < 0.05$ significance level. An attempt was also made to have at least 20 paired samples for each test. This mainly achieved using 20-fold cross-validation.

A.2 Supplementary Results

Supplementary Results on the OSP

Figure A.1 and A.2 respectively illustrate the results that were obtained when SVM-reg and Naïve Bayes were applied to the three OSC datasets. Each point, in these various graphs, represents the average MSE that was obtained in 20 folds of cross validation.

Supplementary Results on Positional Weighting

Figure A.3 and A.4 respectively illustrate the results that were obtained on the Actors and Edmunds datasets by the two weighting functions for uni-tags of sizes 1, 3, 5, 7, 9, 11, 13, 15, and 17. Notice that the Actors dataset generally benefits from the w_α function (i.e. intro+con weighting) but looses with the w_β function (i.e. body weighting). This is inline with the distribution in Figure 5.1 of chapter 5 where sentiment rich words within the Actors documents were mainly concentrated within the first half with a few spikes in the conclusions. A similar conformance is also observed with the Edmunds dataset in that Figure A.4 suggests that the Edmunds dataset generally benefits

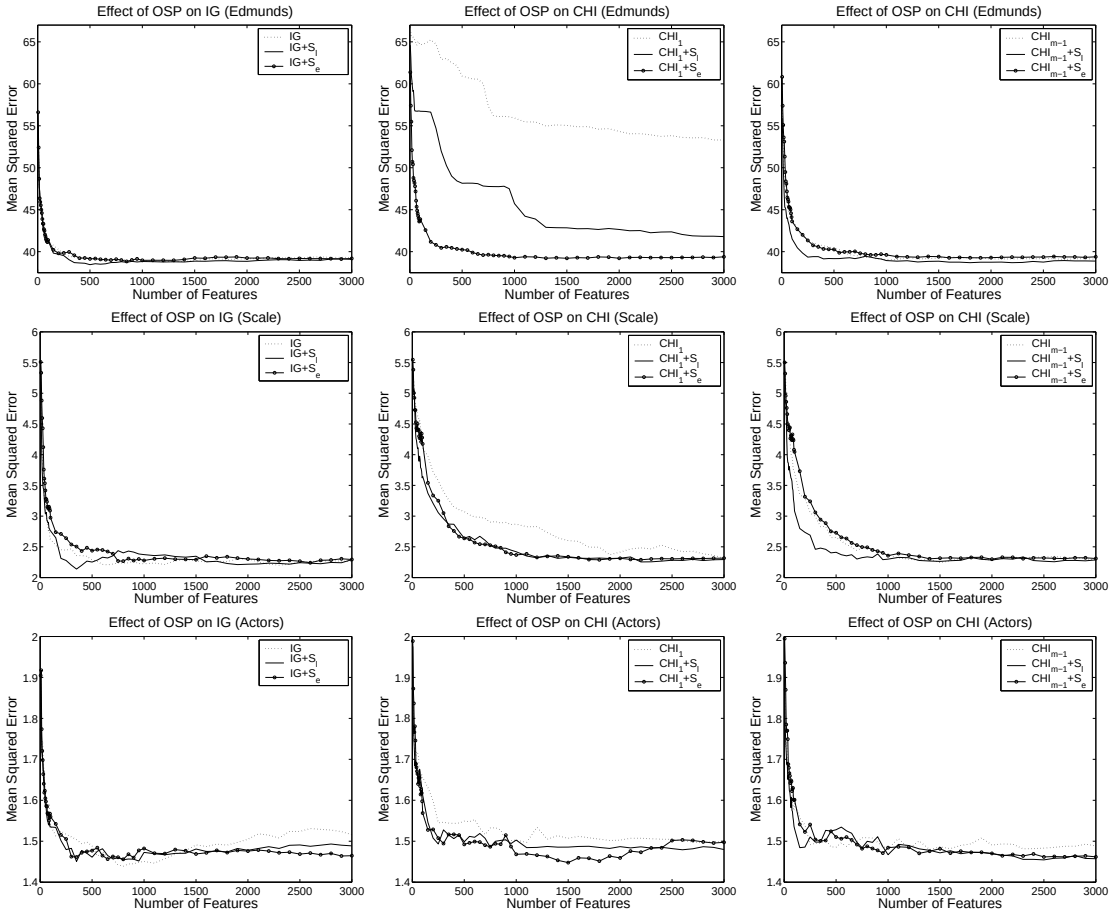

Figure A.1: The Impact of the OSP while using SVM-reg

with the w_β function but looses with the w_α function. This also matches up with the Edmunds distribution in Figure 5.1 of chapter 5 where sentiment rich words of the Edmunds documents were mainly concentrated in the body.

A.3 Opinion Retrieval of Blogs

This section provides an account of the system that was developed in this research to participate in the Opinion Retrieval Task in the Blog Track of the 2007 TREC [Macdonald et al., 2007]. The theme of the task was Opinion Retrieval of Blogs.

A blog (a contraction of the term "Web log") is a Web site, usually maintained by an individual with regular entries of opinions and descriptions of events amongst other things. The key aspect of Blogs that makes them attractive is that they are mainly authored by independent individuals, with

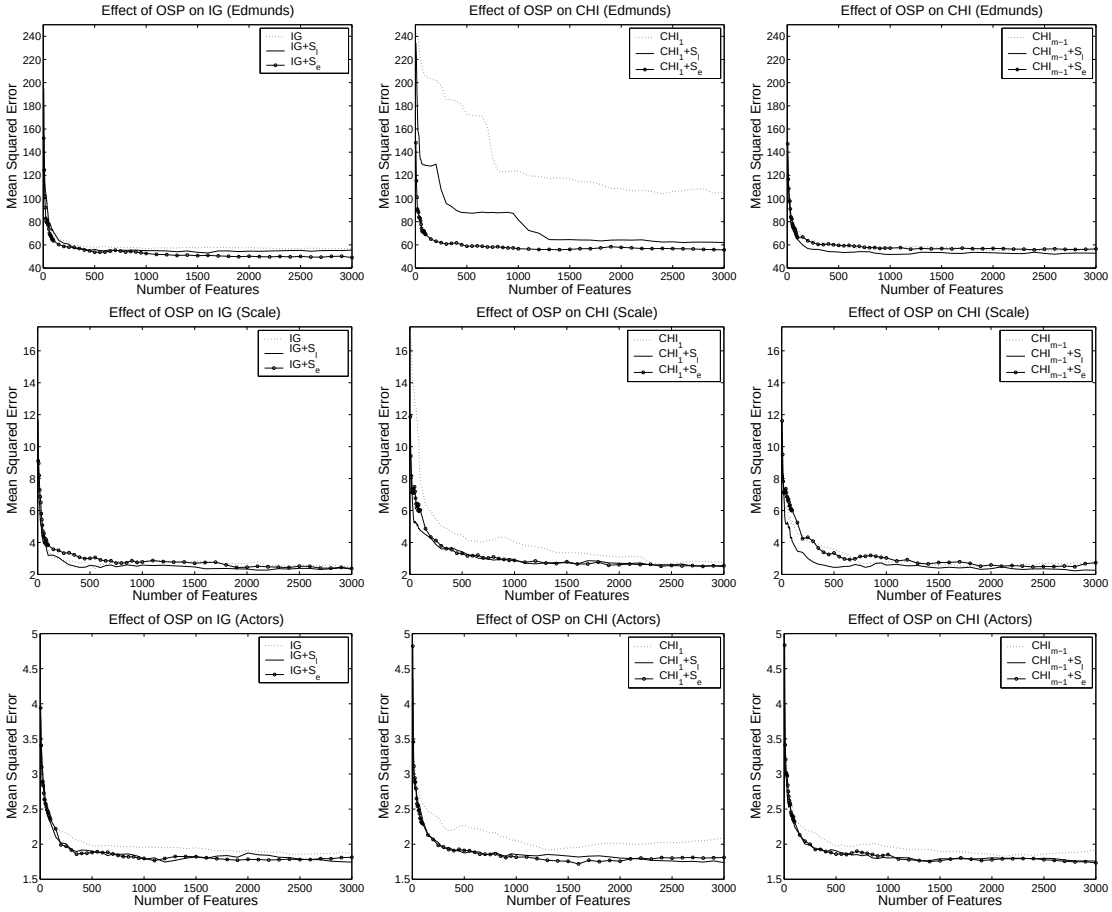

Figure A.2: The Impact of the OSP while using Naïve Bayes

the sole purpose of making their opinions known to the world. Consequently, blogs are highly rich in sentiment and this makes them an invaluable resource for SAoT research.

The Opinion Retrieval Task was first introduced in the TREC of 2006. It basically involves retrieving opinionated documents that are relevant to each of the 50 predefined TREC topics regardless of their opinion orientation. Each retrieved document should, however, be assigned to a real-valued *opinion score* in the range of $[0 \ldots 1]$, where 0 signifies a neutral opinion, whereas 1 signifies an extreme opinion that could be either positive or negative. The collection used in both the 2006 and 2007 Blog Tracks [Macdonald and Ounis, 2006] consists of over a three million blog posts collected over 77 days. It was meant to be a realistic snapshot of the blogosphere and hence offers an excellent test-bed for the study.

The idea that was presented in this research to address the task is composed of two phases that draw from both NLP and IR. The technique basically exploits the context of opinion related

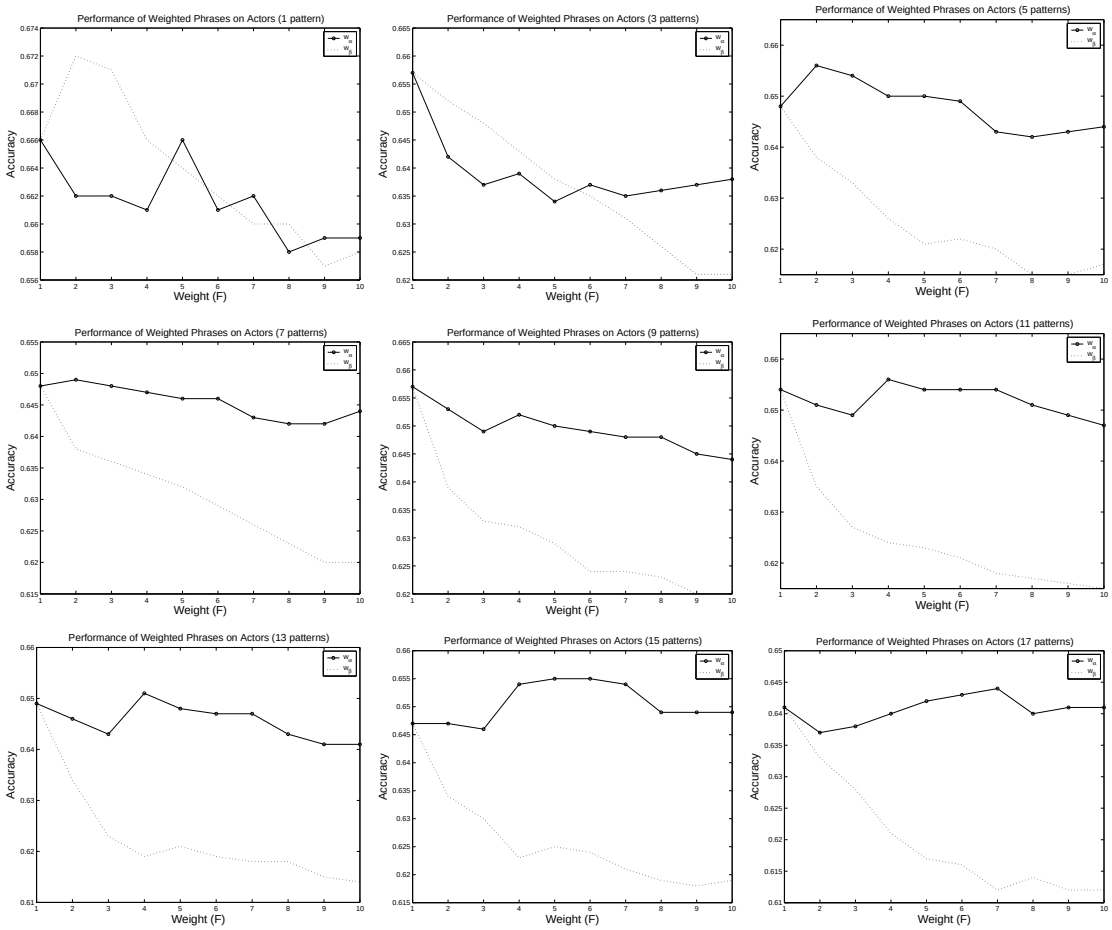

Figure A.3: Performance of the Weighting Functions on the Actors Dataset

language constructs, such as adjectives verbs and adverbs, in order to identify and rank opinionated texts within the collection. These language constructs were chosen primarily because they are commonly associated with opinion rich contexts. For instance, an adjective such as “great” would typically occur within contexts such as “great player,” or “great disaster.”

In the first phase of the procedure, PoS tags are assigned to the features contained in a set of training documents that are labelled with respect to opinion. These documents are then pruned such that only a set of selected words, along with their respective contexts, are retained. Proper nouns are also omitted as they tend to be domain specific [see Turney, 2002]. This results in a corpus with a high precision of opinion rich phrases that are relatively domain independent. The documents of the resultant corpus are then mapped onto a vector space [Salton et al., 1975] after an SVM-reg classifier is trained on the resultant document vectors.

In the second and last phase, a Lucene search engine is used to retrieve all documents, from

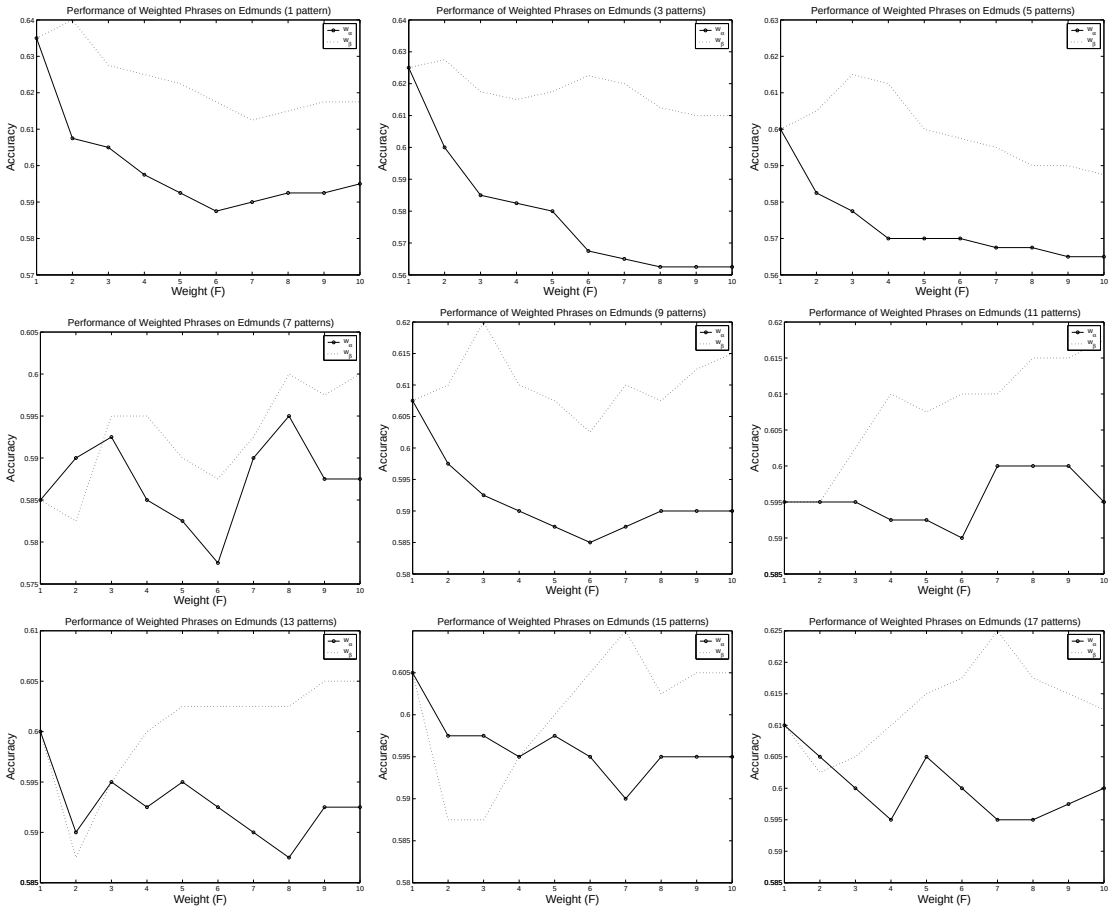

Figure A.4: Performance of the Weighting Functions on the Edmunds Dataset

the TREC Blog collection, that are relevant to the current topic of interest. The SVM-reg classifier is then used to assign each of these documents to an opinion score, and this completes a single TREC run.

A.3.1 The Opinion Retrieval System

A complete overview of the opinion retrieval system has been illustrated in Figure A.5. A crucial resource for this system is the background set of training documents that are labelled with respect to opinion. Each of these labels assumes a value in the ordered set $\{c_1, \ldots, c_n\}$, where $c_1 < \ldots < c_n$ and c_1, c_n respectively represent an extreme negative and positive opinion.

These training documents offer a good estimate of the structure and content of opinion rich texts. However, one limitation with them is that they may not be general enough to suffice as good document examples for each of the 50 TREC topics. To tackle this problem, a procedure that

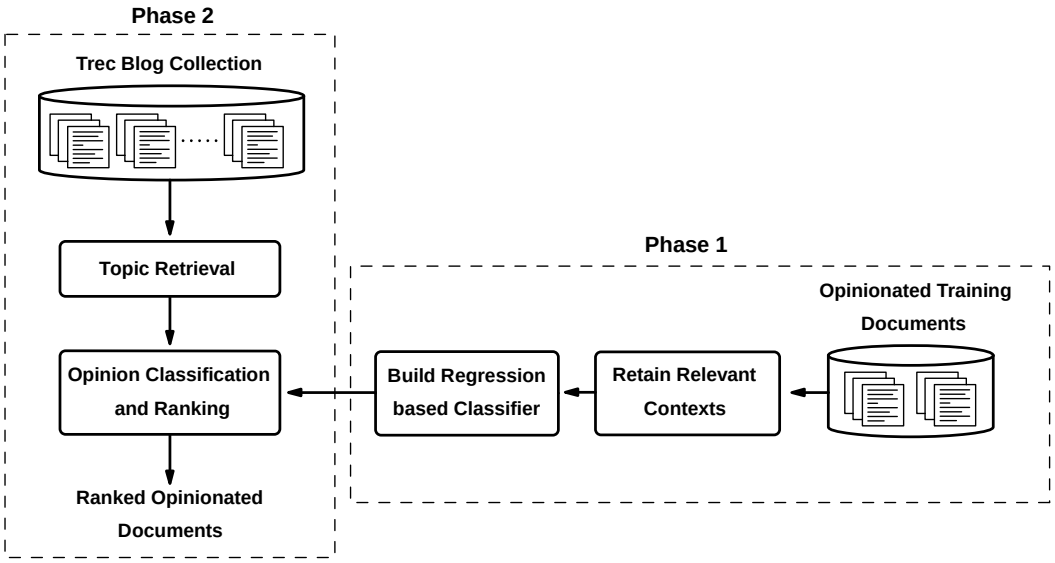

Figure A.5: An Overview of the Opinion Retrieval System.

retains the relevant contexts of language constructs such as adjectives verbs and adverbs is applied. The hypothesis behind this is based on the fact that these constructs act as the transmitters, rather than the objects, of an opinion. Consequently, they would be used in much the same way across various domains. This type of pruning would therefore generalise the opinions expressed within the training documents.

To retain the relevant contexts, PoS tags were first applied to the text within the training documents by using the RASP PoS tagger [Briscoe and Carroll, May 2002]. A context of 10 words was then retained on either side of each word that was tagged as an adjective, verb, or adverb i.e.: JJT, JJ, JJR, VV0, RR, RG, RGA, RGR. Singular and plural proper nouns were, however, not retained as these tend to be domain specific [see Turney, 2002]. Once the training documents were pruned, they were then mapped onto a vector space whose dimensions, or features, were determined by using IG. SVM-reg was then trained on the resultant document vectors.

Finally, given a list of documents from the TREC Blog Corpus that are relevant to a TREC topic, the SVM-reg classifier was used to assign the i^{th} document in this list to a score q_i that assumes a real-value in the range $[c_1 \dots c_n]$. Note, however, that the TREC rules require that the score q_i be mapped onto the range of $[0 \dots 1]$, where 0 signifies a neutral opinion, whereas 1 signifies an extreme opinion that could be either positive or negative. In order to accomplish this, the score of the i^{th} document was mapped to the value $\left|2\left(\frac{q_i - c_1}{c_n - c_1}\right) - 1\right|$ which satisfies the TREC

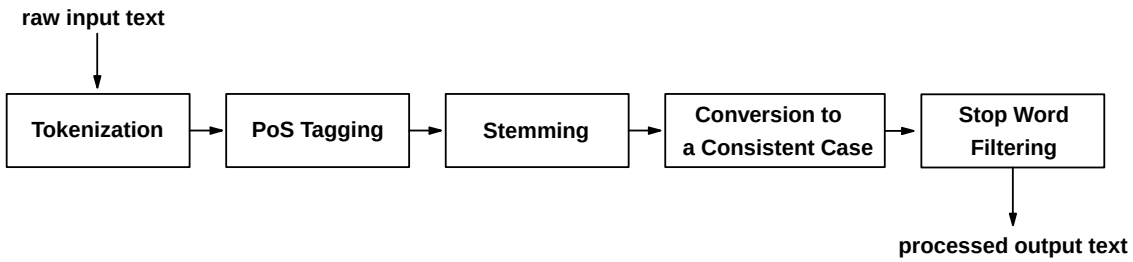

Figure A.6: The Textual Pre-Processing Pipeline

requirement.

A.3.2 Implementation

In order to prepare the TREC Blog collection, the text was first extracted from the initial HTML format discarding all tokens that contained non-printable characters. It was then pre-processed using the sequence of tokenization, conversion to lowercase, stemming and stopword removal (The 50 TREC topics also went through the same pre-processing steps). The resultant collection was finally indexed using the Lucene[1] search engine. This entire procedure lasted 46 continuous days on standard hardware running on an Ubuntu Linux platform.

The second task was to prepare the opinionated training datasets. These were four in number namely: Edmunds with classes $\{1, \ldots, 26\}$, Rateitall with classes $\{1, \ldots, 5\}$, Scale with classes $\{1, \ldots, 8\}$, and the documents that constituted the results of the TREC 2006 polarity task, which had classes $\{0, \ldots, 4\}$. Note, however, only classes 2, 3, and 4 of the last dataset were used. These respectively correspond to a negative, a neutral, and a positive opinion. All four datasets were pre-processed in the similar fashion using the sequence of tokenization, PoS tagging, conversion to lower case, stemming, and stopword removal.

Once the four training datasets were ready, their contexts were pruned as discussed in the previous section. They were then used, in succession, to train SVM-reg classifier in order to classify the documents that were relevant to the 50 TREC topics. The outcome of the four train-classify sessions formed the basis of four of the runs that were submitted to TREC. The fifth run was based on plain relevance retrieval. The following list is a summary of all the five runs that were submitted:

[1] http://lucene.apache.org

Classifier Used	PoS tagging performed?	Datasets			
		Actors	Edmunds	Polarity	Scale
SVM-reg	No	1.4628	40.0806	0.1436	2.2495
	Yes	1.4833	40.1345	0.1418	2.2346
NB	No	2.1058	61.7548	0.1580	2.7946
	Yes	2.1988	61.1400	0.1545	2.7378

Table A.1: The Effect of Part-of-Speech Tagging

1. rgu0: All opinion finding features turned off. Simply a Relevance run.
2. rgu1: Edmunds dataset used as background training data.
3. rgu2: Rateitall dataset used as background training data.
4. rgu3: Scale dataset used as background training data.
5. rgu4: TREC Polarity dataset used as background training data.

Although the official Robert Gordon University (RGU) TREC results were far from being the best, the highest Mean Average Precision (MAP) of 0.2798 improved significantly from the previous years result of 0.0001. Hopefully this trend will continue for successive TREC competitions.

A.3.3 Discussion and Summary

Amongst the five runs that were submitted, the best one achieved a MAP of 0.2798. It also took seventeenth position among all the runs that were submitted by the 20 participants. Although this performance leaves a great deal to be desired, the approach of exploiting the context of adjectives, verbs, and adverbs to identify opinionated text was quite innovative. Future work intends to build upon this approach by investigating the effect of variable sized contexts. Another possible improvement is to employ Case Based Reasoning (CBR) techniques, in conjunction with NLP, to determine the focal point of a context. The hope is that the system will undergo versions of improvement through annual participating in the conference.

A.4 Textual Pre-Processing

This section empirically investigates the effect that each of the textual pre-processing modules in Figure A.6 has on sentiment classification performance. This investigation shall be done on

four datasets: Actors, Edmunds, Polarity, and the Scale Datasets. Note also that tokenization is bypassed as it is compulsory in all experiments.

A.4.1 Part-of-Speech Tagging

Part-of-Speech tagging is the process of marking up the words in a text as corresponding to a particular PoS. This mark-up is based on both the word's definition, as well as its context, i.e., relationship with adjacent and related words in a phrase, sentence, or paragraph. Interpreted this way, PoS tagging would therefore serve as a crude form of word sense disambiguation. For instance, it would distinguish between the different usages of "love" in "I love this movie" (a verb indicating a sentiment orientation) and "This is a love movie" (a noun neutral of sentiment). In order to investigate the effect of PoS tagging on sentiment classification, the four datasets (Actors, Edmunds, Polarity, and Scale) were subjected to a *control* and a *test* experiment. The control did not apply PoS tags to the datasets, whereas the test did. All other conditions were maintained constant. Tagging was performed by using the RASP PoS tagger[2] [Briscoe and Carroll, May 2002] which took on average about 5 seconds to tag each document. The mode of tagging was such that each word was appended with its corresponding PoS tag. For instance, if the word "house" was found to be a common noun then it would be replaced with "house_NN," where NN is the PoS tag that represents common nouns.

Table A.1 illustrates the average MSE results that were obtained over 20 folds of cross validation using both SVM regression (SVM-reg) and Naïve Bayes (NB). For each dataset-column of each classifier, the performance that is significantly better ($p < 0.05$) than the other is shown in bold. Note that PoS tagging did not have any significant effect on performance. As a consequence of both this and the time overheads associated to tagging a corpus, it was opted not to employ PoS tagging as a default pre-processing step.

A.4.2 Stemming

Stemming is related to the study of morphology which deals with the various forms that a given word can assume. A common question in morphology is whether to separate or collapse word forms such as *get*, *gets*, and *got*. Intuitively, performing such a grouping seems like the right

[2]Employs the CLAWS2 Tagset: http://www.comp.lancs.ac.uk/ucrel/claws2tags.html

Classifier Used	Stemming performed?	Datasets			
		Actors	Edmunds	Polarity	Scale
SVM-reg	No	1.4684	39.7385	0.1391	2.3152
	Yes	1.4656	40.2771	0.1438	2.2573
NB	No	2.1778	62.5465	**0.1535**	2.7893
	Yes	2.1098	62.0600	0.1650	2.8334

Table A.2: The Effect of Stemming

action to take as it would stem out redundant words that linguistically refer to the same thing. This grouping process is referred to as *stemming* or *lemmatization*, as it resembles a process of identifying members of a lexeme and replacing them with a representative word.

There has not been much study on the effect of stemming in sentiment analysis. Nonetheless, studies within the Information Retrieval (IR) community have shown that stemming can be quite useful in many types of queries [Hull, 1996]. However, the hypothesis here is that stemming might be slightly harmful to sentiment classification as it may lead to the loss of information. For instance, when groups of terms such as “baddest” (urban lingo for best) and “bad” (standard meaning of dislike) are accidentally grouped into one lexeme. In order to investigate this, the four standard datasets were subjected to a control experiment that did not include stemming, and a test experiment that did. All other conditions were maintained constant.

Table A.2 illustrates the average MSE results that were obtained over 20 folds of cross validation. For each dataset-column of each classifier, the performance that is significantly better ($p < 0.05$) than the other is shown in bold. As was envisaged, stemming had a slightly negative effect on performance. This is clearly evident in the Polarity dataset where the application of stemming significantly lowered the performance of the Naïve Bayes classifier. Nonetheless given that stemming is known to be useful for IR queries, and also that some of the later studies here are based on IR queries, it was opted to apply stemming but only during these studies. Hence, unless specified otherwise, all subsequent experiments in this research do not apply stemming as a default procedure.

A.4.3 Conversion to a Consistent Letter Case

Letter case often contains a great deal of information that can be useful in decoding sentiment. For instance, it can be used to distinguish between a proper noun neutral of sentiment such as

Classifier Used	Type of Letter case	Datasets			
		Actors	Edmunds	Polarity	Scale
SVM-reg	lower	1.4656	40.2771	n/a	n/a
	original	1.4737	40.3622	n/a	n/a
NB	lower	2.1098	62.0600	n/a	n/a
	original	2.0986	61.7360	n/a	n/a

Table A.3: The Effect of using a Consistent Letter Case

"Black" in "James Black" from a sentiment rich adjective such as "black" in "black Monday." Another, perhaps more important, role that letter case plays in this domain is in stressing a point. Clearly the phrase "AWESOME RIDE" would send a more convincing message than "awesome ride." For this reason, the hypothesis held here is that preserving the original letter case may be beneficial to performance. This hypothesis was investigated by subjecting two of the four standard datasets (Actors, Edmunds) to a control experiment that shifted all letters to lower-case, and a test experiment that maintained the original letter case. All other conditions were maintained constant. Note that the Polarity and Scale datasets were not used as they were originally in lower-case.

Table A.3 illustrates the average MSE results that were obtained over 20 folds of cross validation. For each dataset-column of each classifier, the performance that is significantly better ($p < 0.05$) than the other is shown in bold. Note that maintaining the original letter case hurts the performance of SVM-reg but improves that of Naïve Bayes. Nonetheless, none of the differences are statistically significant and hence there is no strong evidence that preserving the original letter case would be beneficial to performance. This is probably because there is too much variety in the manner that capitalisation is used. It was therefore opted to adopt a *consistent letter case policy* in all subsequent experiments of the research.

A.4.4 Stop-Word Filtering

Stop-word filtering is a kind of data set reduction procedure that functions by eliminating tokens that may deem to be redundant in the corpus. It is applied after both PoS tagging and stemming because these two processes depend on contextual information which would be lost as a result of stop-word filtering.

There are at least two main motives for performing stop-word filtering. Firstly, it may lead to an improvement in classifier performance. Secondly, the overall decrease in feature set size can

A	CANNOT	INTO	OUR	THUS	‘
ABOUT	CO	IS	OURS	TO	”
ABOVE	COULD	IT	OURSELVES	TOGETHER	(
ACROSS	DOWN	ITS	OUT	TOO	)
AFTER	DURING	ITSELF	OVER	TOWARD	-
AFTERWARDS	EACH	LAST	OWN	TOWARDS	_
AGAIN	EG	LATTER	PER	UNDER	+
AGAINST	EITHER	LATTERLY	PERHAPS	UNTIL	{
ALL	ELSE	LEAST	RATHER	UP	}
ALMOST	ELSEWHERE	LESS	SAME	UPON	[
ALONE	ENOUGH	LTD	SEEM	US	]
ALONG	ETC	MANY	SEEMED	VERY	:
ALREADY	EVEN	MAY	SEEMING	VIA	;
ALSO	EVER	ME	SEEMS	WAS	@
ALTHOUGH	EVERY	MEANWHILE	SEVERAL	WE	’
ALWAYS	EVERYONE	MIGHT	SHE	WELL	\
AMONG	EVERYTHING	MORE	SHOULD	WERE	—
AMONGST	EVERYWHERE	MOREOVER	SINCE	WHAT	<
AN	EXCEPT	MOST	SO	WHATEVER	,
AND	FEW	MOSTLY	SOME	WHEN	>
ANOTHER	FIRST	MUCH	SOMEHOW	WHENCE	/
ANY	FOR	MUST	SOMEONE	WHENEVER	.
ANYHOW	FORMER	MY	SOMETHING	WHERE	!
ANYONE	FORMERLY	MYSELF	SOMETIME	WHEREAFTER	?
ANYTHING	FROM	NAMELY	SOMETIMES	WHEREAS	*
ANYWHERE	FURTHER	NEITHER	SOMEWHERE	WHEREBY	$
ARE	HAD	NEVER	STILL	WHEREIN	£
AROUND	HAS	NEVERTHELESS	SUCH	WHEREUPON	&
AS	HAVE	NEXT	THAN	WHEREVER	=
AT	HE	NO	THAT	WHETHER	#
BE	HENCE	NOBODY	THE	WHITHER	~
BECAME	HER	NONE	THEIR	WHICH	%
BECAUSE	HERE	NOONE	THEM	WHILE	^
BECOME	HEREAFTER	NOR	THEMSELVES	WHO	
BECOMES	HEREBY	NOT	THEN	WHOEVER	
BECOMING	HEREIN	NOTHING	THENCE	WHOLE	
BEEN	HEREUPON	NOW	THERE	WHOM	
BEFORE	HERS	NOWHERE	THEREAFTER	WHOSE	
BEFOREHAND	HERSELF	OF	THEREBY	WHY	
BEHIND	HIM	OFF	THEREFORE	WILL	
BEING	HIMSELF	OFTEN	THEREIN	WITH	
BELOW	HIS	ON	THEREUPON	WITHIN	
BESIDE	HOW	ONCE	THESE	WITHOUT	
BESIDES	HOWEVER	ONE	THEY	WOULD	
BETWEEN	I	ONLY	THIS	YET	
BEYOND	IE	ONTO	THOSE	YOU	
BOTH	IF	OR	THOUGH	YOUR	
BUT	IN	OTHER	THROUGH	YOURS	
BY	INC	OTHERS	THROUGHOUT	YOURSELF	
CAN	INDEED	OTHERWISE	THRU	YOURSELVES	

Figure A.7: A Standard Stop Word List

result in advantages relating to storage space and computational costs. This study only investigates the first motive. The second one is beyond our scope as both storage and computational efficiency

Classifier Used	Stop-word list used	Datasets			
		Actors	Edmunds	Polarity	Scale
SVM-reg	standard	1.5075	40.7649	0.1438	2.2567
	modified	**1.4656**	**40.2771**	0.1438	2.2573
NB	standard	2.1832	62.3118	**0.1605**	2.8624
	modified	2.1098	62.0600	0.1650	2.8334

Table A.4: The Effect of a Modified Stop-Word List

are issues that are more relevant for applications that are of industrial proportions.

Although stop-word filtering is a widely accepted procedure in text classification, it has had a fair share of criticism. Nigam et al. [2000] for instance, found it to be detrimental to performance hence omitting it from their experiments. There have also been at least two studies in sentiment classification that have also cautioned against the injudicious usage of stop-word list. In one of them, Pang et al. [2002] found tokens such as "!" and "?" to be amongst the most discriminative features in a sentiment rich corpus. Interestingly, these tokens are also typical members of most standard stop-word lists. In the other study, Mukras et al. [2007a] also proposed that standard stop-list members such as "not" should be retained in the corpus as they are often used to negate a sentiment orientation. For instance "not" negates the orientation of the word "good" in "not good."

The hypothesis regarding stop-word filtering is that one should use a specialised stop-word list for the SAoT domain. In order to test this hypothesis, the four standard datasets were subjected to a control and a test experiment. The control employed the standard stop-word list shown in Figure A.7, whereas the test employed a modified stop-word list. This modified list was the same as the standard one but without the following tokens: cannot, not, !, ?, *, $, £, &, =, #, ˜, %, ˆ. These tokens were chosen based on both previous studies [Pang et al., 2002; Mukras et al., 2007a] and intuition.

Table A.4 illustrates the average MSE results that were obtained over 20 folds of cross validation. For each dataset-column of each classifier, the performance that is significantly better ($p < 0.05$) than the other is shown in bold. Note that in 5 out of 8 occasions the modified list was on average more effective than the standard one. In addition to this, amongst these 5 occasions, 2 of them were statistically significant. These results clearly suggest positive benefits in employing the customised list, and hence it was adopted as the default list.

A.5 List of Abbreviations

AvA: All-vs-All

BSA: Binary Search Approach

BSA-KL: BSA KL-distance

BSA-RN: BSA Random

CHI: Chi-Squared Score

DF: Document Frequency

IG: Information Gain

IR: Information Retrieval

MSE: Mean Squared Error

NLP: Natural Language Processing

NSR: Non-Sentiment Rich

OvA: One-vs-All

OSC: Ordinal Sentiment Classification

OSP: Ordinal Smoothing Procedure

PoS: Part-of-Speech

PPS: PoS Pattern Selector

PMI: Pointwise Mutual Information

PMI-IR: PMI Information Retrieval

SAoT: Sentiment Analysis of Text

SOA: Simple Ordinal Approach

SR: Sentiment Rich

SVM: Support Vector Machines

SVM-reg: Regression Based SVM

TREC: Text Retrieval Conference

Blog: Web Log

A.6 Published Papers

Some parts of the work presented in this thesis have appeared in following publications.

1. R. Mukras, N.Wiratunga, R. Lothian, S. Chakraborti, and D. Harper. Information Gain Feature Selection for Ordinal Text Classification using Probability Redistribution. In *Proc. of IJCAI Textlink Workshop*, 2007.

2. R. Mukras, N. Wiratunga, and R. Lothian. Selecting Bi-Tags for Sentiment Analysis of Text. In *Proc. of AI-2007*, pages 181–194, Cambridge, England, 2007. Springer.

3. R. Mukras, N. Wiratunga, and R. Lothian. The Robert Gordon University at the Opinion Retrieval Task of the 2007 Trec Blog Track. In *Proc. of TREC*, 2007.

4. S. Chakraborti, R. Mukras, R. Lothian, N. Wiratunga, S. Watt, and D. Harper. Supervised Latent Semantic Indexing using Adaptive Sprinkling. In *Proc. of IJCAI*, pages 1582–1587. AAAI Press, 2007.

5. A. Orecchioni, N. Wiratunga, S. Massie, S. Chakraborti, and R. Mukras. Learning Incident Causes. In *Proc. of ICCBR TCBR Workshop*, 2007.