

**Automatic Text Categorisation of Racist
Webpages**

Edel Greevy, B. Sc

**A thesis submitted to Dublin City University,
for the degree of
Master of Science**

August 2004

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Science in Computing is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Idul Meeny

Student No.: 96533340

Date: 20th May 2004

Acknowledgements

I would like to express sincere gratitude to my supervisor Prof. Alan Smeaton for his invaluable advice and support throughout this thesis.

I would like to thank my colleagues on the PRINCIP project, in particular Dr. Maggie Gibbon, Dr. Heinz Lechleiter and Dr. Patrick Martin. It's been a fun and enlightening experience to work with such talented people.

Thanks also Neil O'Hare whom I plagued on a few occasions with SVM questions.

I would also like to thank the organisers of JADT-04, the 7th International Conference on the Statistical Analysis of Textual Data for supporting travel to the conference.

Special thanks to my family and friends and to Declan for reading draft after draft after draft of this thesis.

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Our Objective	4
1.3	Thesis Layout	4
2	Automatic Text Categorisation	6
2.1	Defining Text Categorisation	6
2.2	Brief History of Text Categorisation	7
2.3	Applications of Text Categorisation	9
2.4	Text Categorisation and Information Retrieval	13
2.5	The Indexing Procedure	14
2.5.1	Indexing	15
2.5.2	Other Operations	27
2.6	Summary	36
3	Machine Learning	37
3.1	What is Machine Learning?	37
3.2	Training and Test Sets	39

3.3	Methods for Classifier Construction	41
3.3.1	Probabilistic Classifiers	41
3.3.2	Decision Trees	42
3.3.3	Decision Rules	43
3.3.4	Rocchio	43
3.3.5	Neural Networks	44
3.3.6	Example-based Classifiers	44
3.3.7	Regression Classifiers	44
3.3.8	Inductive Rule Learning	45
3.3.9	On-line Learning	46
3.3.10	Support Vector Machine Versus Other Methods	46
3.4	Support Vector Machines	48
3.4.1	About SVMs	48
3.4.2	Generalisation Theory	49
3.4.3	Linear SVM for the Separable Case	54
3.4.4	SVM for the Non-Separable Case	62
3.4.5	SVM for the Non-Linear Case	68
3.5	Summary	72
4	Text Categorisation for Racism on the WWW	73
4.1	Detecting Racism	73
4.2	About the approach	76
4.2.1	PRINCIP Findings	77
4.3	About the Dataset	81
4.3.1	Collecting the Dataset	81
4.3.2	Dataset Statistics	82

4.4	Building the Classification System	83
4.4.1	Building Training and Test Data	83
4.4.2	Learning	88
4.4.3	Classifying	90
4.5	Evaluation	90
4.6	Summary	92
5	Results	94
5.1	Bag of Words	95
5.2	Bigrams	97
5.3	Part-of-speech Tags	100
5.4	Comparing BOW, Bigrams and POS	100
5.4.1	Analysis of Performance in terms of the F1-measure . .	101
5.4.2	Analysis of Performance in terms of Precision	101
5.4.3	Analysis of Performance in terms of Recall	103
5.5	Tuning SVM Parameters	104
5.5.1	Polynomial as a Kernel Function	104
5.5.2	Sigmoid tanh as a Kernel Function	109
5.5.3	Radial Basis Function as a Kernel Function	113
5.6	Summary	117
6	Conclusion	119
6.1	Overview	119
6.2	Which Representation?	122
6.3	Which Classification Tool?	123
6.4	Which Kernel?	124

CONTENTS

vi

6.5 Future Work, Criticisms and Conclusions 125

List of Tables

2.1	Light, Moderate and Heavy Pruning Thresholds used in Fürnkranz [1998]	24
2.2	Contingency Table for Chi-square	33
4.1	Distribution of the parts of speech	80
4.2	Size of the individual datasets	82
5.1	Evaluation of different term weighting measures for BOW	95
5.2	BOW performance	96
5.3	Comparison of a term weighting measure on BOW and bigrams on Set 1	97
5.4	Bigrams performance	98
5.5	POS performance	100
5.6	BOW performance using the polynomial kernel	104
5.7	Bigram performance using the polynomial kernel	105
5.8	POS performance using the polynomial kernel	106
5.9	BOW performance using the sigmoid tanh kernel	109
5.10	Bigram performance using the sigmoid tanh kernel	110
5.11	POS performance using the sigmoid tanh kernel	110

LIST OF TABLES

viii

5.12 BOW performance using the radial basis function as a kernel .	113
5.13 Bigram performance using the radial basis function as a kernel	114
5.14 POS performance using the radial basis function as a kernel .	115

List of Figures

2.1	An example of a POS tagging error	18
3.1	All possible labellings of 3 points in R^2	52
3.2	4 points that cannot be shattered in R^2	53
3.3	The linear, separable case	55
3.4	Support Vectors are positioned on H_1 and H_2	57
3.5	Linear separating hyperplanes for the non-separable case	62
4.1	A racist text	75
4.2	A non-racist text	75
4.3	A document tagged by Xelda in XML format	87
5.1	BOW: Precision and Recall figures for each dataset	96
5.2	BOW: F1-measure scores	97
5.3	Bigrams: F1-measure scores	99
5.4	Comparing F1 scores for BOW and Bigrams	99
5.5	Comparison of F1-measure for BOW, Bigrams and POS	101
5.6	Comparison of Precision for BOW, Bigrams and POS	102
5.7	Comparison of Recall for BOW, Bigrams and POS	103

5.8	Comparison of F1-measure for BOW, Bigrams and POS using Polynomial Kernel Function	107
5.9	Comparison of Precision for BOW, Bigrams and POS using Polynomial Kernel Function	107
5.10	Comparison of Recall for BOW, Bigrams and POS using Polynomial Kernel Function	108
5.11	Comparison of F1 for BOW, Bigrams and POS using Sigmoid tanh Kernel Function	111
5.12	Comparison of Precision for BOW, Bigrams and POS using Sigmoid tanh Kernel Function	112
5.13	Comparison of Recall for BOW, Bigrams and POS using Sigmoid tanh Kernel Function	113
5.14	Comparison of F1 for BOW, Bigrams and POS using Radial Basis Function as a Kernel	116
5.15	Comparison of Precision for BOW, Bigrams and POS using Radial Basis Function as a Kernel	116
5.16	Comparison of Recall for BOW, Bigrams and POS using Radial Basis Function as a Kernel	117

Abstract

Automatic Text Categorisation (TC) involves the assignment of one or more predefined categories to text documents in order that they can be effectively managed. In this thesis we examine the possibility of applying automatic text categorisation to the problem of categorising texts (web pages) based on whether or not they are racist.

TC has proven successful for topic-based problems such as news story categorisation. However, the problem of detecting racism is dissimilar to topic-based problems in that lexical items present in racist documents can also appear in anti-racist documents or indeed potentially any document. The mere presence of a potentially racist term does not necessarily mean the document is racist. The difficulty is finding what discerns racist documents from non-racist.

We use a machine learning method called Support Vector Machines (SVM) to automatically learn features of racism in order to be capable of making a decision about the target class of unseen documents. We examine various representations within an SVM so as to identify the most effective method for handling this problem. Our work shows that it is possible to develop automatic categorisation of web pages, based on these approaches.

Chapter 1

Introduction

1.1 Motivation

Automatic text categorisation is concerned with the assignment of documents to predefined categories and has been successfully applied in many areas that involve the organisation, filing, filtering or routing of documents. These tasks are part of our everyday lives and can be applied to many contexts such as, assigning patents, advertisements or library books into categories, assigning webpages to YAHOO!-style directories or filtering spam. Text categorisation has proven successful for such problems, with results comparable to human evaluation and performance. Such methods can lead to vast improvements in terms of time, manpower and productivity – the same human effort is no longer required as the machine takes over the classification task.

Existing methods for the detection and removal of hate online include the setting up of regulatory authorities. For example the Netherlands has set up the Complaints Bureau for Discrimination; and hotlines exist in EU coun-

tries which allow for potential breaches of legislation to be reported. Sites are investigated and if found to be illegal, are eventually removed. Such solutions are found to be weak because of the fluidity and size of the Internet. Documents originating in the USA, where legislation is most liberal, can be accessed across the globe but belong to another jurisdiction. Technical approaches thus far implemented include Internet Content Filters or Label Bureaus, which simply label sites and filter offensive ones (Internet Content Rating Association¹). Email is typically filtered using regular expressions containing offensive keywords but this approach is unreliable, as it will only filter those emails containing the known keywords and not newer ones. The Safer Internet Action Plan, sponsored by the European Commission, is currently funding various filtering and rating projects including:

- ICRAsafe project will create a system to allow responsible adults to restrict children's access to Internet content that may harm them or which is otherwise considered undesirable by the adult;
- NETPROTECTII is a European tool for Internet access filtering to provide textual filtering in eight European languages;
- PRINCIP aims at the development of a multilingual system for the detection of racism on the Internet.

All project descriptions can be viewed on the Safer Internet Action Plan website.²

Current methods of filtering racism rely heavily on either keywords or the labelling of offensive material. In order to implement successful systems, a

¹<http://www.icra.org/>

²<http://www.saferinternet.org/>

considerable human effort is required, not only in the initial stages of filter construction but also on an ongoing basis as the targets of racism change, the language evolves, existing websites are edited or new websites are added. Given the fluidity of the web, this is one area that may benefit from the application of automatic techniques to text categorisation.

1.2 Our Objective

Our objective is to apply automatic text categorisation techniques to the problem of detecting racism online. We will use Support Vector Machines³, a machine learning method, to build a classifier that will classify texts based on whether or not they are racist. We will talk briefly about the PRINCIP project⁴ and how it helped in our approach to solving this problem using text categorisation techniques. We analyse the problem of detecting racism and highlight the difficulty of this task comparing it to similar problems in the literature. We will investigate different representations of the training data within the Support Vector Machine in order to evaluate the most effective method for detecting racism online.

1.3 Thesis Layout

Chapter 2: Automatic Text Categorisation serves as an introduction to the area of automatic text categorisation (TC). We provide a brief history of the area and list some of the applications of TC. The vari-

³<http://svmlight.joachims.org/>

⁴<http://www.princip.net>

ous steps and methods involved in building an appropriate document representation interpretable by the classifier is core to this chapter. Indexing, stemming, stop-word removal and dimensionality reduction are examined.

Chapter 3: Machine Learning introduces various methods to classifier construction that have been described in the literature. We focus on Support Vector Machines (SVM), as this is the approach we will be using for this thesis.

Chapter 4: Text Categorisation for Racism on the WWW takes a closer look at the problem of detecting racism online. We introduce research that influenced our approach to the problem and the various methodologies used for data collection. We also discuss the processes the data must undergo for the building of the training and test data and explain how the Support Vector Machine is trained and used for classification.

Chapter 5: Results presents the findings of the text categorisation system. We introduce and define each of the representations that will be experimented within this thesis. We compare and contrast bag-of-words (BOW), n-gram and part-of-speech (POS) representations and analyse the effectiveness of the TC system on each representation.

Chapter 6: Conclusion summarises the previous chapters which show that text categorisation is possible for the problem of detecting racism online. We discuss the results of our experiments, highlighting the various advantages and disadvantages of each approach. We discuss future work and experiments to be considered in this area.

Chapter 2

Automatic Text Categorisation

This chapter serves as an introduction to the area of automatic text categorisation (TC). It provides an overview of the subject area giving a brief history of the field and will touch on the various steps involved in building an appropriate document representation which can be fed into a classifier. Indexing, stemming, stop-word removal and dimensionality reduction will be examined in this chapter. This chapter serves as an introduction to the following chapter, which will deal more extensively with machine learning and the building of an inductive classifier and will focus in particular on Support Vector Machines (SVMs).

2.1 Defining Text Categorisation

“Text Categorisation (TC) is the task of assigning predefined categories to free text documents” Yang and Liu [1999]. Texts are assigned to categories based on a likelihood or confidence score that is suggested by a training set of

labelled documents to correspond to each category in the assignment. This confidence ranges between either $\{0, 1\}$ or $\{-1, 1\}$ and in order to arrive at a yes/no decision or a plus/minus figure for the inclusion/exclusion of a document in a category, the confidence score must be mapped onto one of the Boolean values $\{0, 1\}$ or one of $\{-1, 1\}$ using thresholds. Text categorisation is formally described by Sebastiani [2002] as the task of determining an assignment of a value from $\{0, 1\}$ to each entry a_{ij} of the decision matrix where $\{C = c_1, \dots, c_m\}$ is a set of pre-defined *categories*, and $\{D = d_1, \dots, d_n\}$ is a set of documents to be classified. A value of 1 for a_{ij} indicates a decision to assign document d_j the category c_i , while a value of 0 indicates a decision not to assign d_j the category c_i . The problem of assigning a category to a document is made more complex by the fact that sometimes a document may belong to more than one category. This is generally application-dependent and certain constraints must be enforced on the categorisation system in order for it to cope with *multi-labelling*. However in the work described in this project we will be dealing with *single-labelled* documents as in this instance a document will be either racist or not racist and therefore the categories do not overlap.

2.2 Brief History of Text Categorisation

Automatic text categorisation has a history dating back to the 1960s. Up until the mid 1980s the problem of text categorisation was solved by manually building automatic classifiers. Knowledge engineering techniques were used which involved the employment of knowledge engineers and domain ex-

perts to manually define a set of logical rules which encoded the membership rules for each category. These rules were then encoded into a system and used to automatically classify documents into a given set of categories. Rules typically took the form of if (*DNF Boolean formula*) then (*assign category*), meaning that if the document satisfied the condition (*DNF Boolean formula*) then the document was classified under the category (*category*). The downside to this approach is the major human effort that is required in order to build and maintain such a system. Knowledge experts must work alongside domain experts with the aim of formally defining a set of rules. If the domain changes then another domain expert must be employed as a new set of rules is required since no two sets of categories are the same. If the set of categories requires updating then both professionals are once again employed in order to add new rules to the system.

A common and successful example of a rule-based expert system which does document categorisation is CONSTRUE, built by the Carnegie Group (Hayes et al. [1990]) for use at the Reuters news agency. Rules were manually constructed to automatically assign subject categories to news stories and Hayes et al. [1990] reports the system to have done so with a precision and recall of over 90%. However, Sebastiani [2002] pointed out that no other classifier has been tested on the same dataset and that it is unclear whether the dataset selected by Reuters was a random or favourable subset of the whole collection. Apté et al. [1994] agreed that the results of the CONSTRUE system were exceptional but criticised the test set for being relatively sparse compared to the number of possible topics, while Yang [1999] pointed out that adapting the CONSTRUE system to other application domains would

be costly and labour-intensive, a reason which has probably discouraged a lot of people from testing CONSTRUE on other domains.

The 1990s saw a renewed interest in the field of automatic text categorisation for various reasons. The availability of more powerful hardware led to a surge in efficiency and productivity. The explosion of documents available in electronic format and the advent of machine learning approaches to automatic classifier construction also contributed towards reigniting interest in the field. These new approaches of automatically constructing classifiers to perform automatic text classification superseded the earlier and costlier knowledge engineering techniques and provided us with results comparable to human evaluation and performance and savings in terms of manpower, as knowledge experts were no longer necessary. The promise of a future of machines capable of reading, examining and making decisions on the categorisation of free text has also led to a growth in interest in the area and subsequently TC has become a major field of research.

2.3 Applications of Text Categorisation

Text categorisation - the organisation, filing, filtering or routing of documents - is something that is part of our everyday lives and it can be applied to many contexts. The assigning of documents to predefined categories is a task that is required in many domains on an everyday basis, such as the labelling of library books or the assignment of patents into associated categories. Until the introduction of automatic solutions, such work has been

carried out manually. PubMed¹, a service of the National Library of Medicine providing access to over 12 million MEDLINE citations and additional life science journals, spends huge amounts of money each year on human indexers. There is, therefore, a strong justification for automatic or semi-automatic text categorisation. The following examples outline some of the more salient applications of TC.

One of the first uses of automatic text categorisation was automatic *document indexing* for use in information retrieval systems relying on a controlled dictionary. In such a system, each document is assigned one or more keywords or key phrases. These words are used to describe the content of the document and belong to a finite set of words referred to as a controlled dictionary which is often composed of a hierarchical thesaurus, such as the MeSH thesaurus which covers the medical field. Trained human indexers assign keywords or key phrases to a document, thereby making it a costly procedure. By viewing entries in the dictionary/thesaurus as categories, document indexing then becomes a categorisation problem and can therefore be addressed by the more sophisticated automatic techniques. The issue is also closely related to the task of automatic metadata generation of documents in digital libraries. To ease retrieval of books they are tagged according to information (*metadata*) that describe them under headings such as title, author, creation date, document format and so on. Through the use of a controlled dictionary the task of automatically generating metadata may be dealt with in a similar manner to document indexing and may thus be solved using automatic text categorisation methods.

¹<http://www.ncbi.nlm.nih.gov/entrez/query.fcgi>

Classified advertisements for newspapers can be organised into their associated predefined category using TC. This procedure is referred to as *document organisation*. In a printed newspaper, classified advertisements appear under certain categories such as Home, Property, Auto, Travel, etc., which means that prior to publication incoming advertisements must be filed according to the newspaper's list of classified categories. Automatic techniques may be employed to relieve the pressure and time-consuming activity of manual classification.

TC may also be employed to filter or route SPAM on an email server, i.e. *document routing* or *document filtering*. The abundance of SPAM being sent via email servers is a major disadvantage of the information age. According to a report by ePrivacy Group [2003], AOL claims to block 2.4 billion spam emails per day . Since such information is unlikely to be of interest to the average email recipient, normal practice is to delete or filter such emails from the inbox. However, email filters are not very effective unless looking for a specific string or strings but not all SPAM can be detected by simply looking for candidate strings. Using automatic techniques, a system can be trained on a dataset of typical examples of SPAM, which is then used to identify patterns that will detect and filter future SPAM entering the email server or a person's inbox.

TC techniques have been successfully used to automatically populate YAHOO!-like hierarchical directories. YAHOO!-like hierarchical catalogues allow users to refine their search to the particular area of the subject hierarchy of interest to them. Such catalogues guarantee that the user is exposed to documents relating only to a particular category. These catalogues or

directories may be populated using TC, namely by classifying web pages or web sites into one of the several categories that make up the directory.

Some recent applications of TC reported by Dumais et al. [2002] in the 2002 Workshop on Operational Text Classification include:

- Multimedia Categorisation by Thomson Multimedia² and Singingfish³ who categorise multimedia streams based on the metadata associated with each file; spam detection by Elron Software⁴;
- The use of text categorisation to support question-answering systems by the Electronics and Telecommunications Research Institute⁵ in Korea;
- The use of text categorisation to support Northern Light Technology's⁶ search engine where "category assignments are used to group the documents returned by free text search, and to improve the quality of their ranking" Dumais et al. [2002].

There are many more applications of TC, both existing and potential. Text categorisation, either automatic or semi-automatic, can lead to vast improvements in productivity as well as savings in terms of time and human effort.

²<http://www.thomson.net/EN/home>

³<http://www.singingfish.com/>

⁴<http://www.zixcorp.com/>

⁵<http://www.etri.re.kr>

⁶<http://www.northernlight.com/>

2.4 Text Categorisation and Information Retrieval

So what do TC and Information Retrieval (IR) have in common? IR is concerned with the matching of a user's information need, expressed as a query, against a corpus of documents in order to rank documents in the corpus in order of their estimated relevance to the information need. TC on the other hand is concerned with assigning documents to predefined categories. Both are content-based management tasks and for this reason TC relies heavily on the basic machinery of IR, borrowing many of the techniques that are traditionally and primarily associated with IR. There are three main steps involved in the building of a text categorisation system. IR techniques are borrowed and applied to each stage.

Indexing: Indexing is the first step in the building of a text categorisation system. The TC system cannot understand documents in plain text format so they must be transformed into a format amenable to the TC system. Indexing is performed on the initial training corpus and on those documents to be categorised by the classifier when it is in operation. We will discuss indexing in the next section 2.5.

Classifier Construction: Classifiers take the training data as input and learn features of that data so that when presented with unseen data, those learned features will be used to make a decision about the category in which the document is assigned. Some techniques traditionally associated with IR are employed during the inductive construction of classifiers. We will introduce some of the approaches to classifier con-

truction in chapter 3 and talk about Support Vector Machines in more detail.

Evaluation: An evaluation procedure is used to measure the effectiveness of the classifier. Having being trained on the training data, the classifier is presented with test data and the performance of the classifier is evaluated. IR-style evaluative measures are commonly employed, such as precision and recall. The evaluative methods used in this thesis will be discussed in section 4.5 of chapter 4.

2.5 The Indexing Procedure

The first stage in the building of a text categorisation system is the indexing procedure. As text documents are not interpretable by a classifier, an indexing procedure must be applied to the text so as to convert or map it onto an appropriate representation that can be fed into the classification system. Indexing is an important step in text classification as the way in which a document is indexed can have a dramatic effect on the performance of a classifier. Term generation and weighting methods that have been tried and tested in the literature will be discussed in greater detail in section 2.5.1.

A number of other operations can be performed on documents before, during and after indexing, the idea being to reduce the number of index terms in the *term space* and thereby enhance the overall efficiency of the classifier without any loss in effectiveness. We discuss these operations in section 2.5.2.

2.5.1 Indexing

In IR each document is usually represented by a vector of n weighted *index terms*. TC has adopted this approach but differences between the various indexing approaches are reflected by different interpretations regarding what constitutes a term and how weights are measured in a document.

Term Generation

Term generation varies in the amount of linguistic and statistical sophistication that is applied. To form the simplest indexing language, each word can be treated as a feature. This is a common approach and is referred to as the *bag of words (BOW)* approach. However, relationships such as polysemy and synonymy which exist between words, do not make this approach an ideal one and for this reason more complex methods are investigated for the creation of an effective set of *terms* or *features*. For instance, many experiments have been carried out using phrases as indexing terms Lewis [1992], Cohen and Singer [1996], Kelledy and Smeaton [1997], Chandrasekar and Srinivas [1997], Fürnkranz [1998], Mladenic and Globelnic [1998], Tan et al. [2002]. In such cases a phrase is either defined using linguistic information (i.e. the phrase is identified as such according to the grammar of the language of the document) or by using statistical methods (i.e. the phrase is identified as such according to the recurring frequency of a set of words).

There is conflicting evidence regarding the effectiveness and usefulness of the employment of linguistic methods in TC. Lewis [1992] and Smeaton [1997] have found that experiments using more sophisticated linguistic methods do not perform as well as individual words. On the other hand, Fürnkranz [1998]

claims that a richer, more fine-grained representation of the document can provide a learner with valuable information about a document and lead to improvements in performance. The application of linguistic procedures does allow us to express language in a clearer structure in terms of the roles words play in a text and the relationships between words, and this in turn provides a classifier with richer information about a document. Unfortunately much of the research that has employed linguistic methods has not been as effective as hoped. But, despite this, many are still investigating the impact of such methods and Lewis [1992] believes that a combined linguistic and statistical approach may be one solution to the problems of text categorisation. In the following section we will introduce some of the literature that uses linguistic methods and those that use statistical methods for term generation and indexing.

Using Linguistic Information in Term Generation

Various linguistic approaches to generating index terms, a process that can also be part of indexing for TC, have been tried and tested in the literature and some of the approaches will be discussed in greater detail in this section.

Chandrasekar and Srinivas [1997] illustrated how syntactic information can be effective in filtering out irrelevant documents after documents have been retrieved by a search engine. They reported on the performance of two different methods of syntactic labelling, namely Part of Speech (POS) Tagging and Supertagging. POS was performed using an n-gram tagger which used a tagset from the Penn Treebank⁷ and a trigram model to assign

⁷<http://www.cis.upenn.edu/treebank/home.html>

supertags to each word. The difference between the two approaches is that Supertagging contains richer more fine-grained information including sub-categorisation and agreement information about words. Chandrasekar and Srinivas claim that syntactic filtering works; however both training and test sets were very small and so larger datasets may produce conflicting results. Since taggers were used to tag the training and test sets, presumably this system would work best on data similar to that on which the taggers were trained; different data might not be tagged as accurately thereby, increasing the error rate of the taggers. POS taggers are prone to mistakes and do not tag correctly 100% of the time. This can in turn lead to erroneous patterns being created, which could be responsible for the miscategorisation of some documents. Figure 2.1 is taken from a POS-tagged document in the non-racist dataset and illustrates a tagging error. In the sentence *Wear or display the flag*, the VERB *wear* is falsely tagged as a NOUN.

Fürnkranz [1998] illustrated how the use of linguistic phrases as input features can improve precision at the expense of recall. Linguistic phrases were constructed using a system called AUTOSLOG which is an automatic method for extracting patterns from a POS-tagged text. The system is fed noun phrases which are used in the construction of linguistic patterns which are in turn used by the classifier during the categorisation of documents. Fürnkranz et al. tested two classifiers using words as features, phrases as features and words and phrases as features. Inconsistencies were noted in how the data was prepared in the pre-processing stages of the learning stage; this may have had an effect on the differences in performance between the two classifiers. Phrases were shown to be more precise at classifying docu-

```

</lexeme>
<lexeme id="597" start="3032" length="4">
  <surface-form>Wear</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>NOUN</part-of-speech>
      <base-form>wear</base-form>
    </sense>
  </sense-list>
</lexeme>
<lexeme id="598" start="3697" length="2">
  <surface-form>or</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>COORD</part-of-speech>
      <base-form>or</base-form>
    </sense>
  </sense-list>
</lexeme>
<lexeme id="599" start="3700" length="7">
  <surface-form>display</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>V</part-of-speech>
      <base-form>display</base-form>
    </sense>
  </sense-list>
</lexeme>
<lexeme id="600" start="3708" length="3">
  <surface-form>the</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>DET</part-of-speech>
      <base-form>the</base-form>
    </sense>
  </sense-list>
</lexeme>

```

Figure 2.1: An example of a POS tagging error

ments but at the expense of recall. It was noted that phrases were chosen as rules by the classifier less frequently and that a word-based rule was deemed to be the most productive. In fact it was found that few of the phrasal rules featured as the top rules which may be explained by the fact that their classifier sacrifices precision for recall. One of the major drawbacks with the use of syntactic information in indexing and in classification is the computational cost of POS-tagging a document. In general many more extensive tests must be carried out before it can be said that linguistic phrases improve the classification.

Lewis [1992] performed tests on the use of syntactic indexing phrases, clustering of these phrases and clustering of words and found these approaches to be less effective than frequency of occurrence, and in order to be productive, many more phrases were needed by the classifier. Lewis showed word-based features to be more effective. This type of representation outweighs the use of the phrases for now, especially since a greater effort is required to build a feature set of phrases (i.e. POS-tagging, feature extraction). Many pre-processing decisions or tunings (which can have a considerable impact on the performance of the classifier) must be made before the classifier can be tested and evaluated. For example in this experiment, Lewis [1992] uses a statistical model for probabilistic text retrieval. In order to work for text categorisation, the probabilities are converted into a binary output using a threshold. However, such a formula assumes that these estimates are equally weighted or equally consistent across categories and also across documents where in reality, this may be far from the truth. The criteria used to select features, estimate probabilities, cluster terms each contribute towards the

performance of a classifier and so perhaps it is a case of finding the right combination or blend of criteria or methods when employing syntactic information. Lewis believes syntactic information adds valuable information and combined with statistical measures is a potentially promising approach to text categorisation.

Freitag [1997] uses grammatical inference methods together with machine learning methods for the extraction of essential information – such as seminar location, starting time, ending time, speaker – from a collection of seminar announcements taken from online bulletin boards. Freitag treats this problem as a classification problem by training the machine to classify fragments of text into, for instance, one of `location` or `non-location`. For this type of problem the bag of words approach does not suffice. Freitag points out that “it can be effective at identifying the approximate location of the relevant text fragments” but it can have “difficulty identifying the location of the correct field boundaries”. So grammatical inferences are used to add syntactic information about the structure of the type of information that is being sought. For instance, if we want to find the start time of the conference or seminar we would be looking for something of the form (`number, colon, number, number`). Prior to training, the data must be transferred into a format amenable to automatic methods – the training samples are tokenised, tokens are translated into abstract symbols and a grammar is inferred over the abstract sequences. This information is then used in training. Freitag found that this method resulted in improved precision and believes that such a method could prove fruitful for information extraction systems.

Musuyama and Nakagawa [2004] recognise that recent research has re-

ported lower performance of SVM-based text categorisation methods when parts of speech are used as input words and large datasets are used. To improve performance, Musuyama and Nakagawa use a two-step categorisation method with a variable cascaded feature selection to predict a pair of the best number of words and the best POS combination for each category. Different POS combinations and number of words (100, 200, 300, . . . , 1000) are tested so as to identify the best POS combination and the optimal number of features for each category. Mutual information is performed on the extracted words to identify the potential keywords of a category. In this paper, for a category C , categorisation involves two steps – step one extracts words from the POS combination for C and categorises texts into positive or negative. Further classification is performed on the texts in the positive category – this time words from all POS types are extracted, mutual information is performed and the documents are classified a second time. Musuyama and Nakagawa reported higher precision than other practitioners such as Fukumoto and Suzuki [2002] who have conducted similar experiments using POS. Although both experiments were conducted on Reuters corpora, Musuyama and Nakagawa used a different dataset to Fukumoto and Suzuki [2002] – and this may well have influenced performance.

Using Statistics in Term Generation

The following illustrates the kinds of statistics-oriented approaches to the generation of indexing terms that have been explored in the literature.

Mladenec and Globelnic [1998] propose an interesting approach, combining feature generation with feature selection through the use of statistical

methods. What is interesting about this method is the fact that the bag-of-words vector space is enriched with phrasal information (word sequences), meaning the classifier is not relying on the performance of phrases alone. N-gram ($2 > n > 5$) sequences of words were tested, with $n = 3$ proving to be the most effective. Two training corpora were used consisting of 5406 and 1995 web documents. The training corpus was represented as a word-vector, stop words were removed and the vector was pruned, removing all words with a term frequency less than 3. The vector was then enriched by adding all n-gram (where $2 > n > 5$) word sequences with a frequency greater than 3. Feature selection techniques (cf. section 2.5.2) were then applied to the feature set, assigning a score to each feature so as to allow a subset of the highly scoring features to be chosen as the final feature set. Mladenic and Globelnik [1998] did not report on how many features (k) were most effective; it would be interesting to compare different values of k so as to identify the most effective k features. It would also be interesting to investigate whether words or word sequences scored highly. This information is not provided in the published paper and may provide some further insight into the value of n-gram word sequences in text classification. The addition of n-gram word sequences improved the performance of the classifier by 7-8%. The size of the training (5406 and 1995) and test sets (300) were rather small and it would be interesting to test this approach on a more widely used dataset such as Reuters and also by using various other classifiers to see if they outperform the Naïve Bayesian Classifier used in this experiment.

Typically, linear classifiers (based on algorithms such as Bayes and Rocchio (cf. chapter 3) assume that the context of a word w has no impact

on the meaning of w , which is of course not true. For this reason Cohen and Singer [1996] aimed to construct a classifier that allows the context of a word w to affect whether the presence or absence of w will contribute to a classification. In this paper, Cohen and Singer investigated two algorithms each of which have different notions as to what constitutes context. For the RIPPER algorithm, a context of a word w is interpreted as a number of other words that must co-occur with w but their order and location in the document is irrelevant. So the presence of a word w will only influence the prediction of document D into class C if w co-occurs with w_2, \dots, w_k . The Sleeping-Experts system, on the other hand, interprets the context of a word w as consisting of words that occur near w and in a fixed order. So the presence of a word w will influence the prediction of document D into class C if certain words occur in a certain order and near w . Cohen and Singer reported impressive results on numerous datasets (Reuters-22173 corpus, AP titles corpus, TREC-AP titles and the Reuters-21578 corpus). Both algorithms achieved lower error rates than the Rocchio classifier and reportedly performed better on the Reuters corpus than any comparable algorithms that were previously applied to this corpus. It is difficult to directly compare the performance of these classifiers with other similar experiments, but since both RIPPER and Sleeping-Experts were thoroughly tested on several datasets and outperformed the Rocchio algorithm each time, it can be deduced that contextual information is useful in text classification.

Fürnkranz [1998] employed a simple but efficient algorithm, very similar to that used by Mladenic and Globelnic [1998], for the generation of n-grams for use in classification. He investigated the use of n-grams as features on two

datasets (Reuters and Ken Lang’s 20 newsgroup dataset) using the RIPPER classifier. Stop words were removed from the corpora, sentence boundaries were ignored, numbers were converted to the letter *D* and all characters were converted to lower case. An algorithm was used to generate n-grams. Each document represented m set-valued features, one for each n-gram size $1 < m < MaxNGramSize$, i.e., when $m = 3$, for example all 3-grams, 2-grams and 1-grams are included in the feature set. The feature set underwent pruning experiments (light, moderate and heavy pruning) in order to find the most effective feature set size. Pruning was based on a user-specified

Table 2.1: Light, Moderate and Heavy Pruning Thresholds used in Fürnkranz [1998]

	MinTermFrequency	MinDocFrequency
Light	3	5
Moderate	5	10
Heavy	10	20

term and document frequency thresholds, where all sequences that occurred at least *MinTermFrequency* and *MinDocFrequency* in the documents were included. Table 2.1 outlines the thresholds that were used during light, moderate and heavy pruning. Like Mladenic and Globelnic [1998] who used a similar method for the generation of features, Fürnkranz also found that word trigrams actually improved the precision of the classifier. The top ten most frequent features for the Ken Lang dataset interestingly enough consisted of unigrams only, most of which were not indicative of any class. All in all Fürnkranz showed promising results for the use of word sequences in text classification – it was shown that after heavy pruning unigrams seem to be

as effective performance-wise as bigrams generated after moderate pruning.

Another similar reported piece of work investigating the use of bigrams to enhance text classification is that of Tan et al. [2002], who used bigrams in addition to unigrams. Those unigrams that appeared in a significant number of documents were selected and used as seeds for the generation of bigrams. Bigrams were generated and chosen on the basis that at least one of the elements of the bigrams had to be a seed. Further feature selection techniques, namely information gain, term frequency and document frequency, were applied to the feature set to find the best discriminators and reduce the dimensionality of the term space. Tan et al. reported improvements in the performance of the system when unigrams and bigrams were used instead of just unigrams. The classifier was tested on two datasets (Reuters and YAHOO!-Science), each of which reported improvements. For the YAHOO!-Science collection 35.2 of the top 100 features in terms of information gain were bigrams while for the Reuters collection 44.6 were bigrams. Considering there were 1426 bigrams and over 160,000 unigrams in the YAHOO!-Science feature set and only 531 bigrams and about 40,000 unigrams in the Reuters features set, these figures are pretty impressive. Although Tan et al. showed that the use of bigrams enhanced the overall performance of the classifier, some of the categories seemed to benefit from the addition of bigrams much more than others. One reason for this was that some categories were best categorised by unigrams meaning bigrams were not necessary for the categorisation task. Another reason suggested that there was an over-emphasis on some concepts for some classes, thereby leading to a misclassification.

Term Weighting

As previously mentioned, each document in a text categorisation task is usually represented by a vector of n weighted index terms. One of the things that differs among methods of term generation is the way in which a term is weighted. The simplest term weighting technique is the binary approach where 1 denotes presence and 0 denotes absence. Alternatively, the number of occurrences or the frequency can be used to weight a term. Frequency is calculated by dividing the number of occurrences of the term in the document by the total number of words in the document. More complex term weighting methods exist, with weights usually ranging between 0 and 1. The term weighting function (see 2.1) by Salton and Buckley [1988] is a commonly used method.

$$TF * IDF(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|Tr|}{\#Tr(t_k)} \quad (2.1)$$

where $\#(t_k, d_j)$ refers to the number of times the term t_k occurs in document d_j and $\#Tr(t_k)$ refers to the number of documents in the dataset Tr in which the term t_k occurs. This is also known as the document frequency of the term. $TF*IDF$ measures the relevance of a term to a document, i.e. the more a term occurs in a document, the more important it is and therefore more representative it is of the content of a document. This function also takes into consideration the fact that a term is less discriminating if it occurs in many documents. For example function words such *at, the, a, of, at, in*, etc.

occur many times in many documents and are not good content indicators. In order to make weights fall between 0 and 1, they are normalised using *cosine normalisation*.

2.5.2 Other Operations

Removal of Stop Words

Before a document is indexed, the normal procedure in information retrieval and in text classification is to remove stop words. Stop words comprise those words which are neutral to the topic of the document (or query in information retrieval) and would therefore generally contribute very little to the classification of a document. They are often defined by a *stoplist* and include articles, prepositions, conjunctions, pronouns and some high-frequency occurring words. This technique is always performed in IR so as to reduce the number of index terms in a document, to enhance computational efficiency and to minimise the amount of superfluous information in the term space — prepositions, conjunctions etc. do not provide information about a document or help in discerning to which category a document belongs.

Many systems use the same generic stoplist consisting of between 300 and 400 words for English. However, research has been conducted into the generation of domain-specific stoplists by Yang and Wilbur [1996]. Such stoplists are typically much larger than the average domain-independent stoplist so as to make the scaling of categorisation systems more tractable when applied to large amounts of data. In a 1996 JASIS paper, Yang and Wilbur [1996] apply the Wilbur-Sirotkin stop word identification method (Wilbur

and Sirotkin [1995]) to text classification in order to reduce the computational cost without having to trade-off on categorisation effectiveness. The Wilbur-Sirotkin approach aggressively removes words by estimating the importance of a word using relevance information between texts instead of traditional word weighting methods which are only based on word frequencies in a text collection. Yang reported that automated word removal based on corpus statistics resulted in considerable savings in time and memory and improvements in retrieval with respect to precision.

Stop words are not removed in experiments using syntactic information to represent the text, for example Lewis [1992], Chandrasekar and Srinivas [1997]. Such experiments require the presence of all of the original words in a sentence or document in order to assign the correct POS tag. *Term* or *feature extraction* techniques are used to identify the most discriminating and effective patterns. Term extraction techniques will be dealt with later on in the chapter under the heading dimensionality reduction.

Stemming

Stemming is another step performed in IR in order to normalise indexing terms. Stemming involves collapsing words that share the same stem by removing the inflectional ending of the word e.g. *race*, *racism*, *racist*, *racially* all become *rac* which in turn leads to a considerable reduction in the dimensionality of the term space. However, this step is not always included in TC pre-indexing steps, as some have found it to be beneficial and improve performance, while others have not.

A variation of stemming, but an approach that employs linguistic rather

than statistical methods, is lemmatisation. Lemmatisation is a linguistic rather than a statistical task because it is closely related to the identification of the part-of-speech of a word. It involves the reduction of words in a document or corpus to their respective lexemes, thereby allowing a researcher to extract all variants of a lexeme. For the text categorisation task, it is more important to know the root (*lexeme*) of a word than to know the word stem. A lemmatiser automatically disambiguates which root applies to each word in a document or corpus. The main difference between a lemmatiser and stemmer can be illustrated in the following example:

$$\begin{aligned} \text{be (lexeme/root)} &\longrightarrow \textit{is, am, was, were, are, be} \\ \text{be (stem)} &\longrightarrow \textit{be} \end{aligned}$$

A stemmer on the other hand is incapable of making this sort of summarisation. Lemmatisation is therefore more sophisticated and a potentially useful pre-indexing step for text categorisation.

In the UPLIFT⁸ project Kraaij and Pohlmann [1996] have shown that linguistic stemming, also known as lemmatising, can “yield a significant improvement in recall over non-linguistic stemming, without causing a significant deterioration in precision”. In contrast, Krovetz [1993] found that his stemming technique does not result in improvements in performance. There are pros and cons to stemming and lemmatisation and as we have seen these pre-processing steps do not always result in an improvement in performance. Performance improvements may be related to the type of categorisation problem at hand. For instance stemming/lemmatisation may lead to improvements in the performance of a classifier for categorisation tasks that can be described

⁸Utrecht Project: Linguistic Information for Free Text Retrieval

as crispy bounded discourse domains, i.e. a category that is clearly defined by a representative set of keywords which are not used to describe other categories. The classifier does not suffer as a result of the lack of specificity of the terms, i.e. *rac* instead of *racially*, *race*, *aces*, *racist*, as the discourse domain is well defined and crispy bound. As a result, the reduced number of terms leads to a reduction in dimensionality, thereby making the classifier more efficient. For discourse domains that are not crispy bound, for example the detection of racism, my expectation would be that stemming/lemmatisation would not lead to an improvement in performance. Slight inflectional variations of words can influence the orientation of a text. For instance, *racially* is predominantly found in racist texts while *racial* is more likely to be found in anti-racist texts. Stemming would reduce both forms to *rac*, thereby making us unable to make a distinction between the two variations – for that reason stemming/lemmatisation could have a negative impact on the performance of the classifier.

Dimensionality Reduction

In TC, since the number of terms occurring just once in a corpus can be extremely high, efforts are made to try and reduce the dimensionality of the term or vector space from r to $r' < r$. A larger number of term spaces can prove to be quite problematic in TC, as such systems do not always scale well to term spaces with high values. Also, vector spaces with high dimensionality can lead to a problem referred to as overfitting, whereby a classifier is also trained to recognise characteristics of the training data that are not necessary.

A good example of overfitting as provided by Sebastiani [2002] is that of a classifier trained on three examples for the category **CARS FOR SALE**. Two of the advertisements were concerned with the sale of blue cars and as a result a classifier would consider the colour of the car (i.e. blue) to be a characteristic of the category. Therefore classifiers affected by overfitting tend to be exceedingly good at classifying training data but not so effective at classifying unseen data. For these reasons efforts have been made to reduce the dimensionality of a vector space from r terms to $r' < r$. Dimensionality reduction (DR) can be applied either locally or globally. Local DR aims to reduce the number of terms chosen for each category while global DR aims to reduce the number of terms that are chosen across the set of all categories. There are two main approaches to doing DR, namely term selection and term extraction. Both will be discussed in further detail in the next section.

Term Selection

The r' terms are chosen by selecting a subset of the original r terms. The idea is to choose a term set that would yield the least reduction in effectiveness but at the same time result in a reduction in the number of terms in the term space. There are many methods of reducing the dimensionality of a term space; the following briefly introduces some of those methods:

Document Frequency $\#Tr(t_k)$ is one of the more simple methods of dimensionality reduction and refers to the number of documents in the set Tr in which the term t_k occurs. Using this technique it is possible to reduce a term space by a factor of 10.

Information Gain measures the number of bits of information obtained

for category prediction by knowing the presence or absence of a term in a document. Joachims [1998b] used the information gain criterion to avoid overfitting and to reduce the number of irrelevant features.

Mutual Information of a term t , i.e. $MI(t, c)$ is the amount of information gained about t when c is learned. This information can be represented in a two-way contingency table where t is a term and c is the category. X is the number of times t and c co-occur, Y is the number of times that t occurs without c , Z is the number of times c occurs without t and N is the total number of documents. See equation 2.2 (below) for how mutual information is measured.

$$\log \frac{X \times N}{(X+Z) \times (X+Y)}$$

(2.2)

$MI(t, c) = 0$ if and only if t and c are independent. Lewis [1992] used mutual information to assess and rank features for each category

Chi-Square measures the lack of independence between t and c . It is a rough measure of confidence, testing whether or not a difference in frequency reflects a real difference between two texts/categories or just an accident. The null hypothesis i.e. where there is no difference between the two texts/categories, is taken as the expected frequency and the observed frequencies are estimated. A 2×2 contingency table is used

where t is a term and c is the category. A is the number of times t and c co-occur, B is the number of times t occurs without c , C is the number of times c occurs without t , D is the number of times neither c nor t occurs and N is the total number of documents (see Table 2.2 for calculation of chi-square).

Table 2.2: Contingency Table for Chi-square

	t	not t
c	A	C
not c	B	D

If the chi-square value is greater than 3.84 then there is less than a 5% probability that there is no difference between two texts. Unfortunately chi-square is said to be not very reliable for low-frequency terms, as the chi-square distribution can no longer be accurately compared if cells in the contingency table are lightly populated.

Term Strength used a confidence score to measure the strength of a term.

Term strength indicates how important a word is and measures how informative a word is in identifying related documents. As discussed earlier in the chapter, this method was investigated by Yang and Wilbur [1996] in order to remove redundant (non-informative) words and thereby enhance text categorisation. The cosine co-efficiency is used in order to identify document pairs that are similar. Using this information, term strength then measures the likelihood of a term appearing in a related document by estimating the conditional probability of a term occurring in the second half of a pair of related documents given it occurs

in the first half. Scores fall between 0 and 1 with 0 being the lowest strength (i.e. words occurring only in non-related documents) and 1 being the highest value, therefore implying that shared words among related documents are more informative than others. This method is more similar to document frequency than to any of the other methods discussed.

Case Study: Yang and Pedersen [1997]

Yang and Pedersen [1997] tested each of the above term selection methods on two corpora (Reuters and a subset of the MEDLINE corpus) using two learning algorithms – the k nearest neighbour (kNN) and Least Linear Squares Fit (LLSF) algorithms. The effectiveness of the feature selection technique was evaluated by accessing the performance of the classifier on the dataset using precision and recall. They showed that document frequency performed nearly as well as information gain. Since document frequency has a lower computational cost, this approach is favoured over the more computationally expensive information gain. Mutual information was found to perform poorly.

Term Extraction

The r' terms are not chosen by selecting a subset of the original r terms and may not at all resemble the original r terms. Rather the r' terms are obtained through a series of alterations, combinations and transformations of the original r terms.

Term Clustering is a cluster analysis method that involves the grouping of terms which are similar. In some cases similarity between terms is

measured by the degree to which two terms occur in the same document. Term clustering can be used for example to group redundant terms, thereby reducing noise and the dimensionality of the term space and increasing the frequency of assignment of a term. It is also used to cluster terms which are similar, based on a similarity measure such as Cosine. Since phrases have a low frequency of occurrence in documents, it is desirable to apply clustering to phrases so as to reduce the r terms to $r' < r$ and to increase their frequency of assignment. Lewis and Croft [1990] conducted experiments on combining syntactic phrase indexing and term clustering techniques to generate phrase clusters. They felt that this was a complementary approach. Syntactic phrase indexing compensates for the loss in semantics, while clustering aids in remedying the poor statistical qualities of syntactic phrases. Despite this seemingly logical approach, Lewis and Croft showed this method to degrade performance and claimed that the results were poor due to the small size of the corpus he used (110,000 words approximately). The statistical inferiority of phrases (the high number of phrases occurring just once or twice) indicates that a much larger corpus is required for phrasal indexing to be able to prove its worth.

Latent Semantic Indexing (LSI) - Deerwester et al. [1990] tries to overcome the problems associated with word-based methods, as the use of polysemy or synonymous words and phrases in a document can be problematic; Schütze et al. [1995] observe that “if there is a great number of terms which all contribute a small amount of critical information, then the combination of evidence is a major problem for a term-based

classifier”. LSI tries to overcome this by organising information into a semantic structure. Instead of characterising documents according to terms in the document, they are characterised according to the domains within which the terms occur. LSI is typically based on Singular Value Decomposition which is a statistical technique. Sorensen et al. [1997] used a semantic network to represent user profiles and articles in the INFOrmer system so as to overcome the problem of polysemy and the negative effect it has on precision.

2.6 Summary

In this chapter we dealt with indexing in depth. Indexing is the first task involved in building a text categorisation system. It transforms documents, which are typically strings of characters, into a representation interpretable by the classifier. We discussed the various ways in which documents can be indexed, highlighting the different interpretations of what constitutes a term and also the different ways in which terms are weighted. We also presented the various steps that can be taken before, during and after indexing to further reduce the size of the term/feature space so as to make the classifier more efficient without loss in effectiveness.

The next chapter will deal with the role of machine learning in text categorisation, and we will introduce the state of the art and various approaches to classifier construction, focussing on our chosen method, Support Vector Machines.

Chapter 3

Machine Learning

Machine learning techniques are used in text categorisation to build a system that will make a decision about whether a document D belongs to a category C . We call this system a *classifier*. This chapter will introduce the various methods to classifier construction that have been described in the literature. It will focus mostly on Support Vector Machines (SVM), as this is the approach we will be using in this thesis.

3.1 What is Machine Learning?

There is no conventional algorithm for the task of assigning any as yet unseen documents to a predefined category, as no mathematical model of the solution can exist and therefore all we are left to work with in building a classifier are examples. Given a set of examples, we might be able to define input and output values for each given example in the dataset, but we cannot do so for every possible example that exists. It is difficult to generalise from examples

to a set of rules or a fixed algorithm for this process. The relationship between the input documents and the desired output category is often too complex to be captured as an algorithm, and so we turn to the technique of machine learning. A machine is said to learn whenever it changes its structure so as to improve expected future performance.

A classic example of an application of machine learning is the speech recogniser. There exists no algorithm to automatically recognise speech from unknown speakers, i.e. no mathematical model can be implemented in order to recognise a person saying, for example, the word `learn`. For English, for example, we have (or have the potential to obtain) many examples of speech spoken by many different people of different nationalities (English, Irish, American, Australian, Canadian etc). In order to solve the problem of speech recognition we can take a number of examples of different people with different accents saying a particular word and present these examples to the learning machine. The machine can then learn to recognise the word `learn` by examining a number of examples, some of which may be spoken by British men, some by Irish children, some by American women etc. When the performance of a speech recognition machine improves after hearing many examples of people's speech, we can say the machine has learned.

Machine learning can be broadly split into two main areas: *supervised learning* and *unsupervised learning*. In supervised learning the training data used to train the learning algorithm consists of many pairs of input/output training patterns – in other words the machine is given the class or output of an input pattern and tries to learn patterns that would arrive at the expected output. In unsupervised learning the training set consists of input

training patterns only and so the machine is trained without having any prior knowledge of the output classes. The machine learns to adapt based on the experiences of the previous training pattern. In machine learning the output can be *binary*, *multi-class* or *regression*. Binary classifiers have a binary output, i.e. (0, 1) meaning a document either belongs to a class or it does not. Multiclass classifiers allow a document to be categorised into one of a finite number of categories - such classifiers are typically built using multiple binary classifiers. In regression models the output is a real numbered output where a document is given a membership value for each class. In this thesis we will be concerned with supervised learning and binary classifiers only.

3.2 Training and Test Sets

Since any classifier relies heavily on the existence of examples, one of the first things required for us to build a classifier, is a dataset or corpus of test and training examples. The training set should already be classified into the correct categories as the system will learn from this set and make classification decisions based on the training corpus. The training set is usually denoted by equation 3.1.

$$S = \{(x_1, y_1), \dots, (x_n, y_n)\}$$

(3.1)

where n is the number of examples. We refer to x_i as examples or instances and y_i as their labels – Cristianini and Shawe-Taylor [2000]. The test is to test the effectiveness of the classifier. In evaluation, each document is fed to the classifier and the classifier’s decision is compared with an expert’s decision so as to evaluate the performance of the classifier. We refer the reader to section 4.5 in chapter 4 for more on evaluation measures.

Data is crucial to automatic text categorisation as the machine draws on examples to build a classification system. For that reason, a representative dataset is of the utmost importance for the classification of unseen documents. In recent years there has been a surge in the availability of documents available in electronic format. Therefore, appropriate datasets are sometimes already available. For instance, many corpora are freely available and might prove useful for a particular TC problem. TREC, the Text REtrieval Conference¹, encourages information retrieval research on large quantities of text and provides datasets to researchers so that methods can be evaluated on the same data. These datasets are large enough so as to model real problems. Because all researchers work with the same data, the results are not affected by variations in the data but are due only to the effectiveness of the methods and techniques being applied. TREC datasets eventually become available in the public domain so it is not just those involved in TREC experiments that benefit from these large datasets.

In some cases the effort required to collect appropriate documents might not be substantial – as in the case of news story categorisation where documents must simply be saved over a period of time. However, an appropriate

¹<http://trec.nist.gov/>

dataset is not always at hand and, as in our case, must be manually collected before work on the text categorisation system can proceed. In section 4.3.1 we give further information on the methodologies used in data collection for TC of racism online.

In chapter 4 we describe the dataset used in our experiments and the criteria that were used to select that dataset. We also explain the processes the training and test sets must undergo so that the classifier is capable of interpreting the information contained within them.

3.3 Methods for Classifier Construction

Classifier construction is the second stage in the building of a text categorisation system. Many methods, approaches and algorithms exist for the construction of a text classification system. This section will briefly introduce the different approaches that have been investigated in the literature.

3.3.1 Probabilistic Classifiers

Probabilistic classifiers view the classification problem in terms of a probability that a document D of binary or weighted terms belongs to a category C . The probability is calculated by applying Bayes Theorem in equation 3.2.

$$(3.2) \quad P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)}$$

$P(d_j)$ is the probability that a randomly picked document has vector d_j as its representation and $P(c_i)$ represents the probability that a randomly selected document belongs to category c_i . The Naïve Bayes approach is more commonly used and makes the classifier more efficient and effective. The naïve part assumes that words are statistically independent of each other and so word combinations are not used as predictors of a class.

Naïve Bayes is one of the more popular methods for classifier construction and many practitioners have experimented with it in the literature — Lewis and Ringuette [1994], Yang and Liu [1999], Chai et al. [2002], Mladenic and Globelnic [1998], Joachims [1998b], Freitag [1997].

3.3.2 Decision Trees

A decision tree classifier consists of a tree generated as a result of processing a set of training examples where leaf nodes correspond to classes, each non-leaf or internal node is labelled by terms which are used to test an attribute, each branch corresponds to an attribute value representing the weight that a term has in a test document and each leaf node is labelled by a category which is used to assign a classification. A document d is categorised by recursively testing for the weights that the terms have in the representation of d . This step is repeated until a leaf node is reached; the label of this leaf node is then assigned to d (see Lewis and Ringuette [1994], Goller et al. [2000], Joachims [1998b] for experiments using this method).

3.3.3 Decision Rules

The first step involved in the construction of a decision rule classifier is the creation of a dictionary containing the features or attributes that represent individual documents in a collection or domain. A representation maps each individual document in a training set using the dictionary. Each document is assigned a label that denotes which category it belongs to and the objective is to find sets of decision rules or patterns that distinguish one category from the others (see Apté et al. [1994]).

3.3.4 Rocchio

The Rocchio technique is a commonly used method in text categorisation. It is a vector-space method which is also very often used in information retrieval for relevance feedback and document filtering and routing. A prototype vector is created for each category using a training corpus. The prototype vector is also known as the centroid, which is in effect the average of all positive examples for each category. Document vectors belonging to a category are weighted positively and other documents are weighted negatively. The Rocchio classifier rewards the closeness of a test document to the centroid of the positive training examples and its distance from the centroid of the negative training examples (see Goller et al. [2000], Drucker et al. [2001], de Kroon et al. [1996] and Joachims [1998b]).

3.3.5 Neural Networks

A neural network classifier consists of a network of units. Input units represent terms and output terms represent categories. Input and output units are connected by edges that have weights. The weight represents the conditional dependence relations between input and output units. A document d is classified by taking its term weights and assigning them to the input units; the units are then propagated through the network and the value that the output unit takes up determines the categorisation decision (see Yang and Liu [1999]).

3.3.6 Example-based Classifiers

The most widely known case of an example-based classifier is kNN (k nearest neighbour). In a kNN classifier, a document d is categorised under a category c by looking at how the k documents most similar or nearest to d have been classified. The similarity score of each neighbouring document to d is used to determine the category of the neighbouring documents. If a large proportion of them have been classified under c , d is classified under c , otherwise d is not classified under category c (see Yang and Pedersen [1997], Yang and Liu [1999], Goller et al. [2000], Yang [2001] and Joachims [1998b]).

3.3.7 Regression Classifiers

One of the most popular examples of a regression classifier is the Linear Least Squares Fit (LLSF) model introduced by Yang and Chute [1994]. In LLSF, each document d has an input vector consisting of words with weights and an

output vector consisting of the categories of the corresponding document. In order to classify a document, the output vector must be determined. LLSF is computed on the training pair of vectors (input and output) to produce a matrix of word-category coefficients. This matrix measures how likely it is that a word is related to a category. These weights are dependent on the information in the training corpus but LLSF assigns weights in such a way as to minimize error. The matrix aids in the classification of a document as it defines a mapping from a document to a vector of weighted categories. A ranked list of categories is obtained for the input document by sorting these category weights (see Yang and Chute [1994], Yang and Liu [1999], Yang and Pedersen [1997], Yang [1995]).

3.3.8 Inductive Rule Learning

RIPPER (Cohen and Singer [1996], Cohen [1995]) is a rule-learning algorithm based on reduced error rate pruning. Rules are learned by greedily adding one condition at a time using information gain until no incorrect predictions are made on the growing set. The growing set is a randomly chosen collection of training documents and it is used to grow or generate rules. The pruning set is used to refine and simplify rules without decreasing the performance of the rule. Those examples in the training set that are covered by a rule are removed so that new rules can be generated for the remaining examples. In this way at least one rule covers each example in the training set.

3.3.9 On-line Learning

In on-line learning the learner receives one example at a time and gives their estimate of the output before receiving the correct value. The current hypothesis is updated in response to each new example and the quality of learning is assessed by the total number of mistakes made during learning (see Cristianini and Shawe-Taylor [2000]).

3.3.10 Support Vector Machine Versus Other Methods

The methods discussed here have their own pros and cons. For this thesis we will be using Support Vector Machines (SVMs), a method introduced by Vapnik [1995] and implemented by Joachims in *SVM^{light}*.² We have chosen SVMs over other more simplistic methods (Naïve Bayes, Rocchio) for the reasons outlined below.

SVMs are capable of overcoming many of the problems associated with efficiency of training, such as overfitting. The aforementioned methods require that measures such as term selection and term extraction be carried out in order to avoid the problems of overfitting. These measures, also known as dimensionality reduction, discussed in section 2.5.2, must be carried out in order to avoid the curse of dimensionality (cf. section 2) which can lead to a classifier being exceedingly good at classifying documents in the training data but not so good at classifying unseen documents. Therefore, SVMs are capable of generalising well in high dimensional spaces, so SVMs should work well in our domain of application where there is a rich representation of words, bigrams etc. What this essentially means that solutions can always

²<http://svmlight.joachims.org/>

be found efficiently, even for training sets with not only many thousands of examples but also high feature spaces. In other methods, high dimensionality can again lead to overfitting, so that measures must be taken in order to control and restrict the number of features and the number of examples in the training set.

SVMs produce a compact representation of the training data which results in evaluation on unseen input being very fast, thereby making it efficient when it comes to testing. Parameter settings of the SVM can be tuned, which allows for inexpensive cross-validation, for instance, comparing one kernel function against another (cf. section 3.4.5 for more information on kernel functions). Joachims [1998b] points out that “SVMs do not require any parameter tuning, since they can find good parameter settings automatically”. The default settings have proven to be the most effective. According to Joachims [1998b], another advantage of SVMs is their robustness where they have shown “good performance in all experiments, avoiding catastrophic failure, as observed with conventional methods on some tasks”.

Initially, a drawback of the SVM was the complexity of the learning theory and the efficiency of the classifier but this has been rectified and in Dumais et al. [1998] we see training speeds that are comparable to simple learning methods such as Rocchio. Finally, according to Cristianini and Shawe-Taylor [2000], SVMs are a very powerful learning method that “since its introduction has already outperformed most other systems in a wide variety of applications” .

3.4 Support Vector Machines

3.4.1 About SVMs

Support Vector Machines are one of the newer learning approaches to automatic classification, introduced by Vapnik in 1992. SVMs are universal learning machines and are based on statistical methods to minimise the risk of error. The actual decision function, which in this context categorises a document, is implemented from a subset of the overall training points and these points are referred to as *support vectors*. SVMs can be used to directly implement the Structural Risk Minimisation principle, the aim of which, for a given learning task, with a finite amount of training data, is to find the best generalisation performance i.e. minimise the risk of error. This is achieved by finding the right balance between accuracy and capacity.

Generalisation Performance

In SVMs, the performance of a machine is measured by its ability to generalise data where generalisation performance refers to the error rates on test sets or the ability of a hypothesis/function to correctly classify documents not in the training set. As Burges [1998] observes, the best generalisation performance will be achieved if the right balance is struck between the accuracy attained on that particular training set and the *capacity* of the machine.

Capacity

The capacity of a machine is the ability of the machine to learn any training set without error. A machine with infinite capacity is said to be capable of

learning l points regardless of how large l is. Despite this, a machine can have either too much or too little capacity and this affects generalisation performance. A machine with too much capacity causes overfitting (as discussed in chapter 2). Burges [1998] gives a good analogy of what happens when a machine has too much or too little capacity: “A machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before; a machine with too little capacity is like the botanist’s lazy brother, who declares that if it’s green, it’s a tree.”

Overfitting and Underfitting

A classifier that is not sufficiently complex can fail to classify a document correctly. This is referred to as underfitting and usually occurs when the training set is not large enough to recognise all features. Overfitting is when a classifier is too complex or has too much capacity so that it no longer fits the data. Classifiers that overfit the data may recognise noise but cannot make proper classification decisions. Overfitting generally occurs when the training set is too specific so that the machine has learned features that are too specific.

3.4.2 Generalisation Theory

As previously mentioned, text categorisation is formally described by Sebastiani [2002] as the task of determining an assignment of a value from $(0, 1)$ to each entry a_{ij} of the decision matrix where $C = \{c_1, \dots, c_m\}$ is a set of

pre-defined categories, and $D = \{d_1, \dots, d_n\}$ is a set of documents to be classified. Input/output pairings (x_i, y_i) represent the relationship between a document (input) and a class (output). The task of the machine is to learn the relationship between the pairs – this is known as the *target function* – so that when presented with an unseen document the machine can make a decision about the target class of that document. The *decision function* estimates the target function and this function is chosen from a set of candidate functions referred to as *hypotheses*.

N input/output pairings are represented by a vector $x_i \in R_n$, $i = 1, \dots, n$ and the associated truth or class y_i , where y_i is 1 if a document d belongs to category c and -1 otherwise. The task of the machine is to learn the mapping $x_i \rightarrow y_i$ where the idea is to find the best target function of all hypotheses presented. The machine is defined by a set of possible mappings $x \rightarrow f(x, \alpha)$. The functions $f(x, \alpha)$ themselves are labelled by the adjustable parameters α where α represents a weight or bias associated with each training point. For a given input x and choice of α , the machine will always arrive at the same output $f(x, \alpha)$. The target function chosen from all the possible choices is the one that minimises risk and leads to the best generalisation performance. This choice of α generates the trained machine.

Risks of Error

$R(\alpha)$ is referred to as the actual risk. This cannot be computed as it depends on the unknown probability distribution $P(x, y)$ from which the data are drawn. $R_{emp}(\alpha)$ is the empirical risk and is measured by the mean rate of error on the training set for a fixed and finite number of observations or

training sets (x_i, y_i) – in short, it is a measure of the number of training points learned incorrectly. $R_{emp}(\alpha)$ is not a probability distribution but a fixed number for a particular choice of α and for a particular training set (x_i, y_i) .

Risk Bound

If l is the number of documents, the following bound holds in equation 3.3 with probability $1 - \eta$, giving an upper bound on the actual risk of error,

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{(h(\log(\frac{2l}{h})+1) - \log(\frac{\eta}{4}))}{l}} \quad (3.3)$$

h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension and is a direct measure of the capacity of the learning machine. η is an arbitrary constant $0 \leq \eta \leq 1$ that guarantees with probability $1 - \eta$ that the equation holds. The right hand side of the equation is called the risk bound or the VC confidence. It is possible to compute the right hand side of the equation if h is known, but it is not possible to compute the left hand side. The lowest upper bound on the actual risk is achieved by minimising the VC confidence and by minimising the empirical risk $R_{emp}(\alpha)$. A balance must be struck between the right hand side and left hand side of the equation in order to generate a machine with the best generalisation performance. Since the right hand side of the equation is partly expressed in terms of h , i.e. the

VC dimension, minimising h bounds the actual risk of error.

The VC Dimension

VC Dimension (h) is a property of a set of functions $f(\alpha)$. For the case of a two-class classification problem, if a given set of l documents can be labelled in all possible 2^l ways, and for each labelling a member of the set of functions $f(\alpha)$ can be found which correctly assigns those labels, we can say that set of documents is shattered (or correctly classified) by that set of functions. The VC dimension for the set of functions $f(\alpha)$ is defined as the maximum number of training documents that can be shattered by function $f(\alpha)$. If the VC dimension is h then there exists at least one set of h documents that can be shattered, but it in general it will not be true that every set of h documents can be shattered (see Burges [1998]).

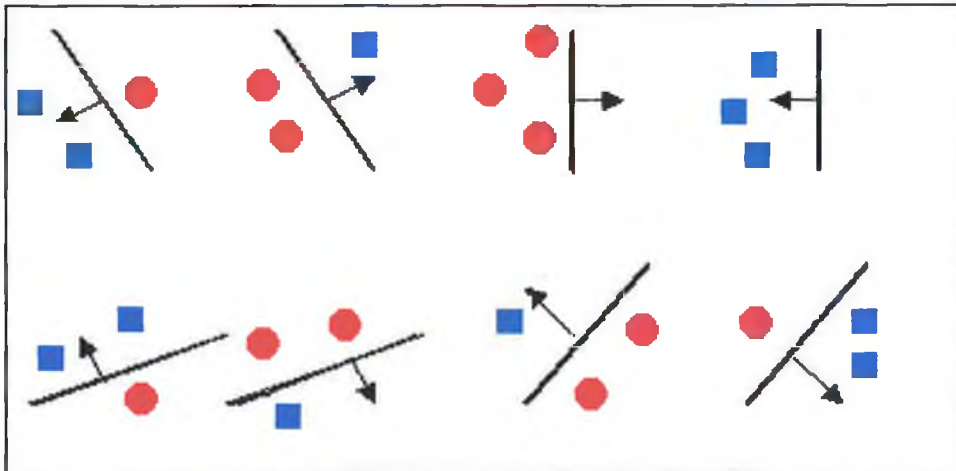


Figure 3.1: All possible labellings of 3 points in R^2

Figure 3.1 illustrates all possible binary labellings (i.e. $2^3 = 8$) of 3 points

in R^2 (two classes). The task of the set of functions $f(\alpha)$ in R^2 is to place points in one of two classes $(1, -1)$, positive or negative. In Figure 3.1, we show how we can shatter three points in R^2 using oriented lines as the $f(\alpha)$ functions. In the hyperplane in Figure 3.1, points on the side in which the arrow is pointing, are labelled 1 (positive). The VC dimension of oriented lines is 3 as the maximum number of points that can be arbitrarily labelled in R^2 is 3.

Consider the labelling of 4 points in R^2 . Figure 3.2 illustrates that oriented lines will not suffice for 4 points in R^2 . Instead hyperplanes must be

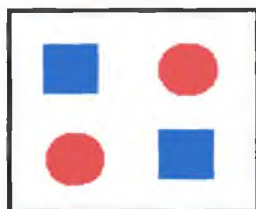


Figure 3.2: 4 points that cannot be shattered in R^2

used for $f(\alpha)$ and the 4 points in R^2 are instead transferred to R^3 . In general, the VC dimension of any set of m points in R^n is $n+1$ (refer to Burges [1998] for theorem proof), therefore the VC dimension of hyperplanes in R^3 is 4.

A learning machine with infinite VC dimension is capable of shattering (classifying) l points, regardless of how large l is. The VC dimension is therefore an indicator of the capacity of a learning machine. Given a selection of learning machines whose empirical risk $R_{emp}(\alpha)$ is zero (i.e. all training points are correctly classified — the linear, separable case), the aim is to choose that learning machine whose associated set of functions has minimal VC dimension as this will lead to a better upper bound on the actual error.

For empirical risk greater than zero that learning machine which minimises the right hand side of equation 3.3 should be chosen.

The Structural Risk Minimisation Principle

The aim of the Structural Risk Minimisation (SRM) principle is to find that subset of the chosen set of functions such that the bound on the actual risk is minimised and an appropriate VC-dimension is chosen. Both the $R(\alpha)$ and $R_{emp}(\alpha)$ depend on a particular function chosen by the training procedure, whereas the VC-confidence depends on a chosen class of functions; these values are used to calculate the risk bound. In order to find the best subset of functions, either h or a bound on h must be computed. This can be done by training a series of machines. The entire set of functions is divided into nested subsets with decreasing capacity – for each subset either h or a bound on h is computed. This can be done by training a series of machines, i.e. one for each subset. The goal of the training is to minimise the empirical risk $R_{emp}(\alpha)$ for a given subset. The subset of functions whose sum of empirical risk and VC confidence is minimal is chosen as the trained machine – this is the machine that gives the best trade-off between the risk of error and the capacity of the machine.

3.4.3 Linear SVM for the Separable Case

The simplest case for the operation of an SVM classifier is that of linear support vector machines trained on separable data. The training data is completely separable – all training samples are learned correctly and the data can be separated by a straight line. Figure 3.3 illustrates what we mean

by both linear and separable.

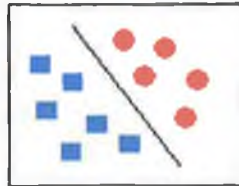


Figure 3.3: The linear, separable case

The training data is labelled as follows for (x_i, y_i) , where $i = 1, \dots, l$, $y_i \in \{1, -1\}$, $x_i \in \mathbb{R}^d$. A hyperplane separates the positive examples from the negative examples; this is also known as the separating hyperplane. The points x which lie on the separating hyperplane (H_0) satisfy equation 3.4.

$$w \cdot x + b = 0$$

(3.4)

where w is normal to the hyperplane and is a vector which defines a direction perpendicular to the hyperplane, b is a bias and varying this value moves the hyperplane parallel to itself. These two parameters, w and b , control the function and must be learned from the data. $\frac{|b|}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin and $\|w\|$ is the Euclidean norm of w . The shortest distance from the separating hyperplane to the closest positive or negative point is d_+ or d_- . The margin of the separating hyperplane is therefore the distance of the closest negative point from the separating

hyperplane ($d-$) plus the distance of the closest positive point from the separating hyperplane ($d+$) or $\frac{2}{\|w\|}$ (see Burges Burges [1998]).

$$H_1 : x_i \cdot w + b \geq 1 \text{ for } y_i = 1$$

(3.5)

$$H_2 : x_i \cdot w + b \leq -1 \text{ for } y_i = -1$$

(3.6)

For the linear separable case the support vector algorithm looks for the separating hyperplane with the largest margin: a large margin minimises the risk of error. The set of labelled training points are said to be linearly separable if there exists a vector w and a scalar b such that the inequalities in equations 3.5 and 3.6 are valid for all elements of the training set (see Cortes and Vapnik [1995]).

$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i$$

(3.7)

Equations 3.5 and 3.6 can be combined into the inequality in 3.7. For any

point actually lying on H_1 or H_2 the equality $y_i(x_i \cdot w + b) - 1 = 0$ holds. These points are termed support vectors (see Figure 3.4).

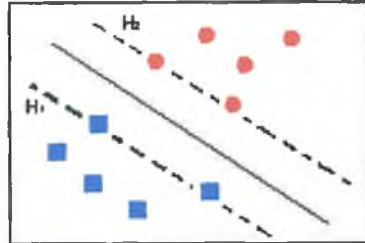


Figure 3.4: Support Vectors are positioned on H_1 and H_2

Maximising the Margin

The optimal hyperplane is the one that maximises the distance between H_1 and H_2 (see Figure 3.4). This is done using a standard optimisation technique referred to as a Lagrangian formulation. The objective function $\frac{1}{2}||w||^2$ is to be optimised subject to the set of constraints in equation 3.7. For constraints of the form $c_i \geq 0$, the constraints are multiplied by positive Lagrange multipliers and subtracted from the objective function. The Lagrange multipliers are unconstrained for constraints of the form $c_i = 0$. This gives the primal Lagrangian formulation in equation 3.8.

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1}^l \alpha_i \{y_i(x_i \cdot w + b) - 1\} \quad (3.8)$$

where α_i are positive Lagrange multipliers, ($i = 1, \dots, l$) and one exists for each of the inequality constraints x_i in 3.7. L_P is maximised subject to the constraint that the gradient of L_P with respect to w and b vanish and also subject to the constraints $\alpha_i \geq 0$ (see Burges [1998]). From that, the following conditions in equations 3.9 and 3.10 apply to that the vector w which determines the optimal hyperplane can be written as a linear combination of the training vectors (see Cortes and Vapnik [1995]).

$$w = \sum_i \alpha_i y_i x_i \quad (3.9)$$

$$\sum_i \alpha_i y_i = 0 \quad (3.10)$$

Equation 3.9 is the result of requiring that the gradient of L_P with respect to w and b vanish. The vector w that determines the optimal hyperplane can be written as a linear combination of training vectors as in equation 3.9. These values can be substituted into the primal equation to give the dual formulation in equation 3.11.

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

(3.11)

L_P and L_D arise from the same objective function but with different constraints. The optimal solution for support vector training for the separable, linear case is found by maximising L_D or minimising L_P with respect to α_i , subject to the constraints in equation 3.10 and the positivity of α_i with the solution given by 3.9. In this formulation of the problem in equation 3.11, the training data will only appear in the form of dot products between the vectors; this feature allows us to generalise the training procedure to the more complex non-linear case. There are Lagrange multipliers α_i for every training point, and in the solution any training points for which the Lagrange multiplier, α_i , is greater than zero are called support vectors (see Burges [1998]). These points lie on one of the hyperplanes H_1 or H_2 and are closest to or actually on the decision boundary. All other training points lie on either side of H_1 or H_2 and have $\alpha_i = 0$ (meaning the equality in 3.7 holds). These points are of great importance to the training set as they are used to obtain the final solution — and therefore their removal would affect the solution. This gives a sparse solution, as only a subset of the original training points are used to calculate the final solution.

The Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker (KKT) conditions play a central role in both the theory and practice of constrained optimisation. The KKT conditions are satisfied at the solution of any constrained optimisation problem. These conditions hold for all support vector machines, since the constraints are

always linear and because the problem for SVM is convex. Therefore the KKT conditions are necessary and sufficient proof that a proposed solution is correct. This means that solving the SVM problem is equivalent to finding a solution to the KTT conditions, i.e. equivalent to finding values for w , b and α which satisfy these conditions.

The KKT conditions for the primal problem for the linear, separable case as outlined above can be stated in 3.12 as:

$$\begin{aligned}
 \frac{dL_P}{dw} &= w - \sum_i \alpha_i y_i x_i = 0 \\
 \frac{dL_P}{db} &= - \sum_i \alpha_i y_i = 0 \\
 y_i(x_i \cdot w + b) - 1 &\geq 0 \\
 \alpha_i &\geq 0 \\
 \alpha_i(y_i(x_i \cdot w + b) - 1) &= 0
 \end{aligned}$$

(3.12)

To reiterate, if all of the above conditions hold, this implies we have a necessary and sufficient proof that w , b and α are the solution, or in other words, we have a classifier. The last of this set of conditions is called the Karush-Kuhn-Tucker complementary condition. The general form is expressed as equation 3.13:

$$\alpha_i g_i(x) = 0$$

(3.13)

where $g_i(x)$ is the set of inequality constraints. Once equation 3.9 is calculated, this value can be substituted into the complementary condition. Any of the support vectors can be used for x_i in this equation. By using appropriate values for α_i and y_i , b can be calculated.

Testing Phase

In the test phase of the development of a classifier, the task is to determine on which side of the decision boundary a given test pattern lies. The decision boundary is the hyperplane lying halfway between H_1 or H_2 – in the test phase test points are pulled either side of the decision boundary – in either a positive or negative direction. This is determined by equation 3.14:

$$f(x) = \text{sign}(w \cdot x + b)$$

(3.14)

or by substituting equation 3.9 we get:

$$f(x) = \text{sign}(\sum_i \alpha_i y_i x_i \cdot x + b)$$

(3.15)

In the equation 3.15, the decision surface appears as a dot product between the training points. The dot product can be seen as a similarity measure and is important when it comes to adapting the SVM model to the non-linear case. The decision function essentially measures each of the test points against each of the support vectors (that chosen subset of the training set lying on one of H_1 or H_2). Positively labelled support vectors drag the result in a positive direction depending on the similarity and the weight α attached to each of the support vectors. If a test point is most similar to the positive support vectors, then it will be classified positively. The same holds for the opposite case.

3.4.4 SVM for the Non-Separable Case

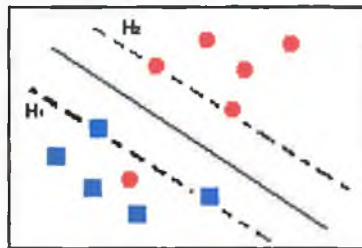


Figure 3.5: Linear separating hyperplanes for the non-separable case

The problem of finding the linear separation of non-separable data with the smallest number of misclassifications is NP-complete. Such data cannot be separated by the above algorithm, as it assumes that a separating hyperplane exists such that $w \cdot x + b = 0$. This equality cannot be satisfied when

the data is not separable. To cater for the more common non-separable case, the constraints in equations 3.5 and 3.6 must be relaxed. The “algorithm works by adding misclassified positive training examples or subtracting negative ones to an initial arbitrary weight vector” (Cristianini and Shawe-Taylor [2000]). Positive slack variables $\xi_i = 1, \dots, l$ are introduced in the constraints giving:

$$x_i \cdot w + b \geq +1 - \xi_i \text{ for } y_i = +1$$

(3.16)

$$x_i \cdot w + b \leq -1 + \xi_i \text{ for } y_i = -1$$

(3.17)

$$\xi_i \geq 0 \quad \forall i$$

(3.18)

Then, equations 3.16, 3.17 are combined to give the inequality in equation 3.19:

$$y_i(w \cdot x + b) \geq 1 - \xi_i$$

(3.19)

The slack variable is a measure of how much the constraints are broken. An error occurs if $\xi > 1$, meaning all points for which $\xi > 1$ will be misclassified. Points with a value between 0 and 1 will be classified correctly. Points with such values fall inside the margin. All other points that are classified correctly will have a ξ_i of zero. The total error therefore equates to $\sum \xi_i$ and can be seen as an upper bound on the number of training points classified incorrectly. See Figure 3.5 for a graphical explanation. This value will feature in the primal objective function. In the linear separable case the objective function to be minimised was $\frac{1}{2}\|w\|^2$ whereas for the non-separable case it becomes:

$$\frac{1}{2}\|w\|^2 + C(\sum_i \xi_i)^k$$

(3.20)

C is a parameter defined by the user and a larger C implies a higher penalty for training errors. The first term in 3.20 is the same as for the linear separable case - it acts as a bound on the VC dimension. The second term acts as a penalty for the number of training errors or a measure of the empirical risk. Note: in the linear separable case we did not have this term, as the empirical risk was equal to zero. In order to find the solution a balance between the VC dimension and the empirical risk must be found and this is the task of the Structural Risk Minimisation principle.

The dual formulation of the problem appears as in 3.21. Neither the ξ_i nor the Lagrange multipliers appear in the dual formulation.

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x_i \cdot x_j \quad (3.21)$$

The only difference is that the α_i now have an upper bound of C (3.22). Note that C is a parameter that is chosen by the user. The higher C the less tolerance there is for error. We wish to maximise L_D subject to two constraints in equations 3.22 and 3.23:

$$0 \leq \alpha_i \leq C \quad (3.22)$$

$$\sum \alpha_i y_j = 0 \quad (3.23)$$

The solution for w is given by equation 3.24 where N_S is the number of

support vectors.

$$w = \sum_{i=1}^{N_S} \alpha_i y_i x_i \quad (3.24)$$

As with the case for separable data, Karush-Kuhn Tucker conditions in equation 3.25 are required in order to solve the primal Lagrangian function for the non-separable case.

$$L_P = \frac{1}{2} \|w\|^2 - \sum_i \alpha_i (y_i (x_i \cdot w + b) - 1 + \xi_i) + C \sum_i \xi_i - \sum_i \mu_i \xi_i \quad (3.25)$$

where μ_i are the Lagrange multipliers. Lagrange multipliers are introduced to keep the slack variables ξ_i positive. L_P is maximised by differentiating with respect to w , b and ξ . The KKT conditions for the primal problem are therefore:

$$dL_P/dw_v = w_v - \sum \alpha_i y_i x_{iv} = 0 \quad (3.26)$$

$$dL_P/d_b = -\sum \alpha_i y_i = 0$$

(3.27)

$$(3.28) \quad dL_P/d\xi_i = C - \alpha_i - \mu_i = 0$$

$$(3.29) \quad y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0$$

$$(3.30) \quad \xi_i \geq 0$$

$$(3.31) \quad \alpha_i \geq 0$$

$$(3.32) \quad \mu_i \geq 0$$

$$\alpha_i \{y_i(x_i \cdot w + b) - 1 + \xi_i\} = 0$$

(3.33)

$$\mu_i \xi_i = 0$$

(3.34)

Equations 3.33 and 3.34 are the Karush-Kuhn-Tucker complementary conditions and these can be used to determine the value of b . Equations 3.28 and 3.34 can be combined to show that $\xi_i = 0$ if $\alpha_i < C$. By taking any training point for which $0 < \alpha_i < C$, b can be computed using equation 3.33 (though it is best to take the average over all training points).

3.4.5 SVM for the Non-Linear Case

The methods we use for the classification of linear, separable data and non-separable data cannot be applied to the classification of non-linear data. By mapping the data from the input space to a higher dimensional feature space, linear separation becomes much easier, thereby allowing the same linear techniques previously discussed to be applied to the data. The term *features* represents the quantities introduced to describe the data, while the task of choosing the most suitable representation is referred to as *feature*

selection, and the space onto which the data is mapped is called the *feature space*. A number of different techniques for selecting the best representation of data exist.

Typically it is best to use the smallest possible set of features that still convey the essential information contained in the data. This is known as *dimensionality reduction*. As the number of features grows it can become computationally expensive and consequently infeasible to solve the classification task. Also, generalisation performance can diminish when faced with too many features, so that the learning machine is prone to overfitting. This phenomenon is sometimes referred to as the *curse of dimensionality* (refer to Cristianini and Shawe-Taylor [2000]).

Mapping the data to a higher dimensional space makes it easier to separate the data with a linear decision function if the target function is a quadratic polynomial. In the training problem the data appears in the form of dot products, $x_i \cdot x_j$ and, if we map the data to a higher dimensional space then the training algorithm would only depend on the data through dot products in H , i.e. on functions of the form $\Phi(x_i) \cdot \Phi(x_j)$. The set of hypotheses to be considered will be functions of the type:

$$f(x) = \sum w_i \Phi_i(x) + b \quad (3.35)$$

where $\Phi : X \rightarrow F$ is a non-linear map from the input space to some feature

space. To recap, w is the vector that determines the optimal hyperplane. Recall that the solution for w is given by $w = \sum_i \alpha_i y_i x_i$. In the test phase, the decision rule can be evaluated using inner products between the test point and the training points. This can be done by computing the following:

$$f(x) = \text{sign}(\sum_i \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b)$$

(3.36)

where x_i are the support vectors. Working in a higher dimensional space is not always a feasible solution to the problem, due to the curse of dimensionality and because it can be computationally expensive. To overcome this, a function referred to as a *kernel function* is used to map the data to a high dimensional space without having to actually create that feature space, i.e. $K(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$; this avoids the computational problems associated with high dimensional spaces and also makes it possible to use infinite dimensions efficiently. In other words, we would only need to use K in the training algorithm and would never need to explicitly even know what Φ is (see Burges [1998]). By working in a feature space where Φ defines the mapping, we replace $x_i \cdot x_j$ with $K(x_i \cdot x_j)$ everywhere in the training algorithm such that we do not explicitly have to create the mapping $\Phi(x_i) \cdot \Phi(x_j)$. In doing so, we allow the data to be dealt with in the same way as the previously mentioned linear, separable case in section 3.4.3. The only extra computational expense involved is that of computing the *kernel function*.

Kernel Functions

The kernel function provides a shortcut to the above problem. Instead of creating the feature space from the original input space, computing the inner product in that feature space and then finding a way of computing that value in terms of the original inputs, the kernel function computes the inner product in the input space without having to explicitly define the feature space.

The choice of kernel function determines whether the resulting SVM is a polynomial classifier, a two-layer neural network, a radial basis function machine or some other learning machine (see Burges [1998]). Given a function K , using Mercer's Theorem, it is possible to verify that it is a kernel (see Cristianini and Shawe-Taylor [2000] for proof). The set of kernels is closed under some operations. If K and K' are kernels then:

- $K + K'$ is a kernel,
- cK is a kernel, $c > 0$,
- $aK + bK$ is a kernel for $a, b > 0$,
- Complex kernels can be made by combining simpler kernels according to specific rules,
- Kernels can be made from features, e.g. the polynomial kernel and the string kernel,
- They can be made from probabilistic generative models by transforming them into similarity functions.

Gaussian RBF (radial based function), polynomial, sigmoidal and inverse multiquadric kernel functions are the most commonly used kernels (see Müller et al. [2001]) and can be defined as follows:

$$\text{Gaussian RBF: } K(x, z) = \exp \frac{-\|xz\|^2}{c} s$$

$$\text{Polynomial: } \langle x, z \rangle^d$$

$$\text{Sigmoidal: } \tanh k_x \cdot y - \delta$$

$$\text{Inverse multiquadric: } \frac{1}{\sqrt{(\|xy\|^2 + c^2)}}$$

3.5 Summary

In this chapter we introduced and discussed machine learning and the role it plays in text categorisation. We introduced various approaches to classifier construction, highlighting the advantages and disadvantages of each. Support Vector Machines were dealt with in greatest detail as this is the method which we will be using to deal with the text categorisation of racist documents on the Internet.

In the next chapter we will discuss the issues related to text categorisation for identifying racism online.

Chapter 4

Text Categorisation for Racism on the WWW

In this chapter we introduce the issues related to detecting racism online. We describe the dataset that is used in our experiments, and the methods used to collect the dataset. We describe our approach to text categorisation for racism and evaluation measures that will be applied in assessing the effectiveness of the TC system.

4.1 Detecting Racism

Automatic text categorisation has proven to be successful for topic-based problems such as news story categorisation. This is based on the classification of articles into categories where there are little or no similarities between the content words of each class. Typically, such problems are clear cut and there is little doubt as to which class a document belongs. For instance, for news

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW74

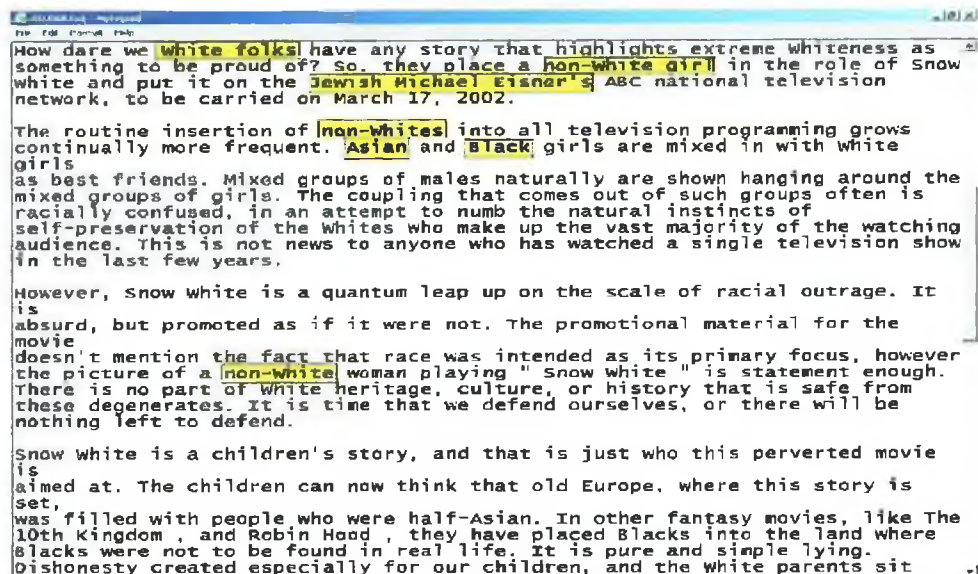
story classification, news items are put into categories such as sport, politics, finance, etc. Topic-based documents are identified on the presence/absence or frequency of certain strings that are more characteristic of one class than another. For example strings such as *ISEQ*, *investment fund*, *interest rate* are likely to be found in financial articles but not in sporting ones.

The problem of detecting racism is quite different to topic-based problems. Racist documents target potentially any race or group of people – *Jews*, *Arabs*, *Muslims*, *African-Americans*, *Africans*, *Aborigines*, *asylum seekers*, *refugees*, *Hispanics*, *Asians*, *Protestants*, *Catholics*, *Turks* etc. Thus detecting racism could be considered partially a topic-based problem – for instance, find all documents containing the word *Jew* and those documents that are racist towards Jews are a subset of all the documents that are about Jews. The difficult task is in finding what discerns racist documents apart from the rest. In Figures 4.1 and 4.2 a number of words (*white*, *black*, *non-white*, *African American*, *Jewish*, *Asian*) have been highlighted which could indicate that both texts are about race. Of course, they may also be racist but we cannot decipher this based on these keywords alone as other factors come into play.

In a recent paper Grilheres et al. [2004] describe experiments carried out during the European Research Project NetProtect II ¹, a project similar to PRINCIP, in that it also aims to filter harmful webpages in order to protect children. However, the NetProtect project attempts to filter documents promoting drug consumption, documents containing recipes for home-made bombs or explosives and documents promoting violence and pornography.

¹<http://www.net-protect.org/>

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW75



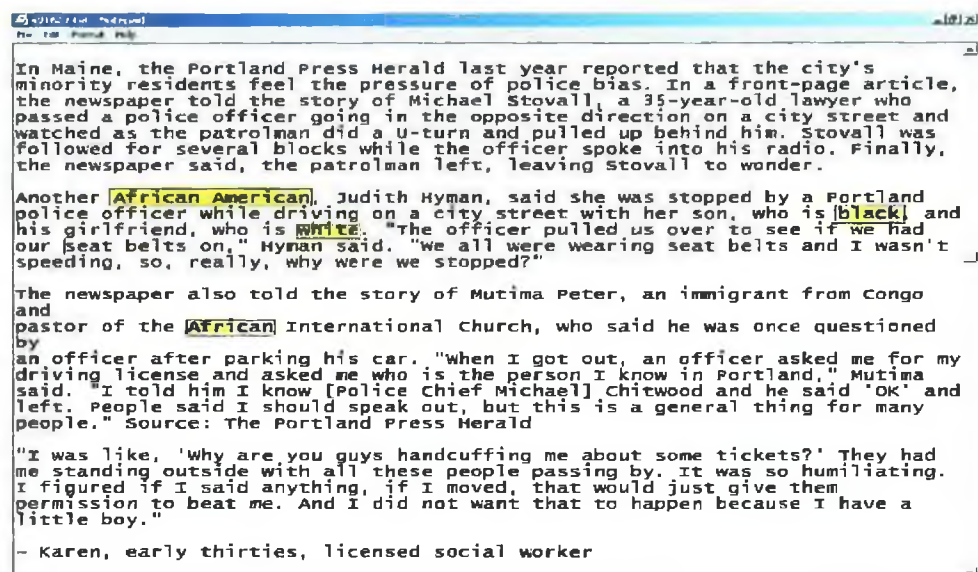
How dare we white folks have any story that highlights extreme whiteness as something to be proud of? So, they place a non-white girl in the role of Snow White and put it on the Jewish Michael Eisner's ABC national television network, to be carried on March 17, 2002.

The routine insertion of non-whites into all television programming grows continually more frequent. Asian and Black girls are mixed in with white girls as best friends. Mixed groups of males naturally are shown hanging around the mixed groups of girls. The coupling that comes out of such groups often is racially confused, in an attempt to numb the natural instincts of self-preservation of the whites who make up the vast majority of the watching audience. This is not news to anyone who has watched a single television show in the last few years.

However, snow white is a quantum leap up on the scale of racial outrage. It is absurd, but promoted as if it were not. The promotional material for the movie doesn't mention the fact that race was intended as its primary focus, however the picture of a non-white woman playing "snow white" is statement enough. There is no part of white heritage, culture, or history that is safe from these degenerates. It is time that we defend ourselves, or there will be nothing left to defend.

Snow white is a children's story, and that is just who this perverted movie is aimed at. The children can now think that old Europe, where this story is set, was filled with people who were half-Asian. In other fantasy movies, like The 10th Kingdom, and Robin Hood, they have placed Blacks into the land where Blacks were not to be found in real life. It is pure and simple lying. Dishonesty created especially for our children, and the white parents sit

Figure 4.1: A racist text



In Maine, the Portland Press Herald last year reported that the city's minority residents feel the pressure of police bias. In a front-page article, the newspaper told the story of Michael Stovall, a 35-year-old lawyer who passed a police officer going in the opposite direction on a city street and watched as the patrolman did a U-turn and pulled up behind him. Stovall was followed for several blocks while the officer spoke into his radio. Finally, the newspaper said, the patrolman left, leaving Stovall to wonder.

Another African American, Judith Hyman, said she was stopped by a Portland police officer while driving on a city street with her son, who is black and his girlfriend, who is white. "The officer pulled us over to see if we had our seat belts on," Hyman said. "we all were wearing seat belts and I wasn't speeding, so, really, why were we stopped?"

The newspaper also told the story of Mutima Peter, an immigrant from Congo and pastor of the African International Church, who said he was once questioned by an officer after parking his car. "when I got out, an officer asked me for my driving license and asked me who is the person I know in Portland," Mutima said. "I told him I know [Police Chief Michael] Chitwood and he said 'OK' and left. People said I should speak out, but this is a general thing for many people." Source: The Portland Press Herald

"I was like, 'why are you guys handcuffing me about some tickets?' They had me standing outside with all these people passing by. It was so humiliating. I figured if I said anything, if I moved, that would just give them permission to beat me. And I did not want that to happen because I have a little boy."

- Karen, early thirties, licensed social worker

Figure 4.2: A non-racist text

Using a search engine such as Google to find documents about the history of bombs or explosives erroneously retrieves documents containing recipes for home-made bombs. This classification problem is, in a way, similar to the PRINCIP problem in that keywords and phrases which may appear to be racist are also found in non-racist discourse, meaning keywords alone are not reliable indicators of whether or not a document is racist. Grilheres et al. [2004] combined classifiers using images, URLs and the most pertinent n keywords. They did not rely on keywords alone for classification and found the combined classifier approach led to better filtering performances than single classifiers. The main difference between the NetProtect problem and the PRINCIP problem is that NetProtect is primarily a topic-based problem whereas detecting racism goes beyond the topic. Detecting racism is more similar to the problem of subjectivity classification as described by Finn et al. [2002] and Finn and Kushmerick [2002]. These researchers perform classifications based on the subjectivity of a text by trying to identify whether a document reports facts or the opinions of the author. Like us, they are not really concerned with topic and are trying to make a classification based on something which is orthogonal to the topic. Detecting texts on the basis of whether they are racist or not is really about discerning the author's opinion or attitude in relation to the topic.

4.2 About the approach

In this thesis we build several text categorisation systems based on different representations of the data. We will analyse the performance of three

representations – the bag-of-words (BOW), bigrams and part-of-speech tags (POS). Support Vector Machines will be used to build the classification systems using each of the representations. We will compare and contrast these representations in the system for the purpose of identifying the most effective method for the detection of racism on the WWW.

All three representations have already been tried and tested, both in IR and in text categorisation – Finn et al. [2002], Chandrasekar and Srinivas [1997], Tan et al. [2002], Finn and Kushmerick [2002], Fürnkranz [1998], Kelledy and Smeaton [1997], Lewis [1992], Mladenic and Globelnic [1998]. Although they have been examined extensively in the context of other domains and different classification problems (i.e. problems related to topic and content as opposed to attitude or opinion), limited experiments related to attitude or opinion have been conducted and to our knowledge no experiments have been carried out on the detection of racism online.

Finn et al. [2002] and Finn and Kushmerick [2002], whose work on subjectivity classification is closely related to our own work, found the BOW approach to outperform POS tags. Also, the results of the research conducted during the PRINCIP project provided further justification for our approach in building a text categorisation system.

4.2.1 PRINCIP Findings

Our own experiments in the PRINCIP project (see Gibbon and Greevy [2003], Lechleiter and Greevy [2003], Martin [2003a,b]) revealed there to be significant differences at lexical, collocational and syntactic level between racist and non-racist texts.

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW78

Lexical Level

We identified lexical items that appeared equally consistently in each dataset. We were able to identify those lexical items that occurred 30% more often in racist texts. Modals, adverbs and truth claims were among this list, *e.g. must, never, once, ever, same, very, course, fact, white, race, nation*. The use of modals, representing the taking of absolute positions, and the use of argumentation structures such as truth claims like *fact* or *of course* are characteristic of racist discourse. For instance:

Least of all **should** we place an alien cult of anti-Aryanism ahead of the survival of Our Race.

We **must** understand that when he follows the semite pied piper he endangers himself and all that which encompasses his race from the beginning of time.

For the Jews, as foreigners, certainly should have nothing from us; and what they have certainly **must** be ours.

Our people **must** realize the dangers of amalgamation and arise to condemn all the individuals and organizations, white, black and mixed, who are devoting time and effort to bring about a mongrelized America.

Colored people are different from Caucasian people. This is an obvious **fact** and the differences between these races often, **in fact** go beyond skin color.

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW79

The greatest Race to **ever** walk the earth, dying a slow death.

The fact that the color line is being broken down, mongrelization is taking place and has taken place, is an abomination against Nature.

Such linguistic features are indicative of the discourse of racist language (see Gibbon and Greevy [2003], Lechleiter and Greevy [2003]). These results confirmed for us that, although the same lexical items may appear in both racist and non-racist texts, differences in frequencies of such items exist. We found certain non-content words which are part of everyday language use appearing significantly more often in racist discourse and their over-use can in itself be an indication of the nature of the text and of the stance taken by the author.

Collocation Level

Our studies revealed that words which appear in racist discourse can also appear in many other discourses. For instance *over-breeding*, a word encountered frequently in racist discourse, also appears in texts about breeding horses. For this reason, we looked beyond keywords and started to look at word sequences and context. This study revealed to us many collocations which are consistently prevalent in racist texts, e.g. *our own kind, white civilisation, white survival, only Jews, our country* were encountered significantly more often in the racist corpus (see Gibbon and Greevy [2003], Lechleiter and Greevy [2003], Martin [2003a,b]).

The results of our linguistic studies showed that it may be possible to

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW80

separate texts based on context and that is one of the reasons we train the Support Vector Machine on 2- and 3-word sequences.

Syntactic Level

The corpus was tagged using XeLDA², a suite of linguistic tools made available to us by Xerox. The distribution of different POS categories was investigated.

Table 4.1: Distribution of the parts of speech

	Racist	Non-racist
ADJ	8.89	8.14
ADV	4.61	3.49
NOUN	21.14	22.40
VERB	14.79	12.18
OTHER	50.57	53.78

The results (in Table 4.1) show there to be differences of between 1-3% across the board. These differences may seem insignificant and rather small, but in order to put these figures into context, the size of the samples must also be taken into consideration. It is interesting to note that the racist corpus contains more adjectives and fewer nouns making the adjective-noun ratio .42 for the racist and only .36 for the non-racist. This tells us that racist discourse contains more qualifiers and that the larger number of nouns in the anti-racist corpus may be indicative of a difference in register.

We trained a Support Vector Machine on the POS information to see what effect this representation has on the categorisation system.

²Xerox Linguistic Development Architecture: <http://www.xrce.xerox.com/competencies/past-projects/platforms/xelda.html>

4.3 About the Dataset

To conduct experiments in the PRINCIP project, a dataset of over 2 million English words and about 1000 documents was collected. This same dataset was used to train the Support Vector Machine which will allow for the identification of racist documents. The dataset consists of an equal number of racist and non-racist documents which were downloaded from the WWW.

4.3.1 Collecting the Dataset

When building the dataset during the PRINCIP project a combination of approaches was used to avoid circularity, to target a diverse collection of documents from different domains, different countries and also to target different groups.

Yahoo!³ and Google directories were browsed under categories containing social and cultural texts. A list of potentially racist keywords and phrases was constructed with the help of the League of Human Rights in Belgium.⁴ Current affairs provided useful clues for the building of the list, as did studying research on racist discourse by Dijk [1987] and Wodak and Reisigl [2001]. The list of candidate keywords and phrases was submitted to search engines such as Google⁵ and AlltheWeb⁶ and the results of the search query were classified into one of either racist or non-racist.

Given the topology of the web, we assumed that like attracts like which meant that by following hyperlinks within webpages we would discover more

³<http://www.yahoo.com>

⁴<http://www.liguedh.org>

⁵<http://www.google.com>

⁶<http://www.alltheweb.com>

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW82

documents of a similar nature. It followed that downloading hyperlinks proved a particularly useful method in building the datasets. We refer the reader to work by the Princip Partners [2002] for further information.

4.3.2 Dataset Statistics

The dataset contains 1000 documents with 500 racist and 500 non-racist documents. These documents were collected using the methods described in 4.3.1. For further information on these methods see work by the Princip Partners [2002].

Table 4.2: Size of the individual datasets

	Set 1	Set 2	Set 3	Set 4
Training set size	200	400	600	800
Test set size	60	100	150	200

The 1000 documents were split into training and test sets. No criteria were defined for this task. Instead, the documents for the training and test sets were chosen at random thereby minimising bias. Each training and test set contains an equal number of racist and non-racist documents.

In order to analyse the impact of differing sized datasets on the system, we split the dataset accordingly as in Table 4.2. In the following chapter we will refer to set 1, set 2, set 3 and set 4.

4.4 Building the Classification System

In this thesis we used Support Vector Machines as the machine learning method to build a text categorisation system. We are using *SVM^{light}*,⁷ which is an implementation of Vapnik's Support Vector Machine (Vapnik [1995]). This package is freely available for scientific use and can be downloaded online.

When downloaded and installed, the *SVM^{light}* package contains two executables:

svm_learn is the learning module. The training data is passed to the learning module; the SVM learns the training data and outputs a model which is used to help classify new unseen data.

svm_classify is the classification module. This uses the model output during learning to help classify new unseen data.

4.4.1 Building Training and Test Data

In section 2.5 we examined indexing procedures and processes and the building of representations interpretable by the classifier. We have already spoken about the need for training and test data in section 3.2. In this section we examine the role of training and test data, including the building of these datasets and their transformation into appropriate representations in the context of this thesis.

The training data is used by the learning module to identify features of each class. In classification mode these features are then used to assign a

⁷<http://svmlight.joachims.org/>

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW84

document to a class. The test data consists of those documents yet to be classified by the system. The classification module assigns a class to each document.

All documents in both training and test data to be classified by the system must be indexed, i.e. transformed into a representation that can be understood by the Support Vector Machine. All training/test data are represented in one file where each line represents a document and takes the form of:

< target > < feature >:< value > ... < feature >:< value >

where

< target > = +1 | -1

< feature > = int

< value > = boolean | int | float

White space separates **TARGET** and the **FEATURE:VALUE** pairs. **TARGET** represents the target class of the document where +1 is assigned to documents which belong to the class (i.e. documents that are racist) and -1 is assigned to documents which do not belong to the class (i.e. documents that are non-racist). In our case each **FEATURE** is a number representing either a word, bigram or POS tag in the dataset. **FEATURE:VALUE** pairs must be ordered by increasing value of feature number. **VALUE** represents the degree to which the feature is present in the document, which could be a boolean value i.e. 0 representing absence or 1 representing presence. Alternatively, it could be an integer value representing the number of occurrences in the document or a floating number which is essentially a value representing the

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW85

weight of the feature in the document. We refer the reader to section 2.5.1 for further information on term or feature weighting.

A document might therefore be represented as follows:

$$-1 \ 1 : 0.59 \ 4 : 0.7 \ 12 : 0.5 \dots$$

where -1 specifies a negative example for which feature number 1 has the value 0.59, feature number 4 has the value 0.7, feature number 12 has the value 0.5 and so on.

BOW

In this representation, each word is treated as a feature. We did not use stop lists or perform stemming or lemmatisation. Two separate experiments were conducted to evaluate the performance of the classifier, where the *value* is taken to be

1. the number of occurrences of the feature in the document
2. the frequency of the feature in the document

Frequency is calculated by dividing the number of occurrences of the feature in the document by the document length.

Bigrams

We also look at bigrams at word level. For instance, for the following sentence

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW86

“Our race is our nation”

the bigrams would be:

‘Our race’ / ‘race is’ / ‘is our’ / ‘our nation’

Since frequency produced better results for the BOW approach than the number of occurrences, we used frequency as a means of weighting terms in the documents, i.e. *value* represented the frequency of the bigram in the document.

We wrote a Java program to create the appropriate input for the building of the training data.

Parts of Speech

The dataset was tagged using XeLDA – Xerox Linguistic Development Architecture. Xelda outputs the tagged document in XML format (see Figure 4.3 for sample output). It is therefore necessary for the document to undergo some pre-processing to extract the tags so that the documents can be transformed into a representation interpretable by the classifier. We wrote a Java program to parse the XML document and extract the POS tags from the tagged text.

The Xelda disambiguation tagset contains 70 tags so the number of features is small in comparison to BOWs and bigrams. In this representation each tag is treated as a feature.

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW87

```
<lexeme id="705" start="4214" length="7">
  <surface-form>healing</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>NOUNING</part-of-speech>
      <base-form>healing</base-form>
    </sense>
  </sense-list>
</lexeme>
<lexeme id="706" start="4222" length="3">
  <surface-form>and</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>COORD</part-of-speech>
      <base-form>and</base-form>
    </sense>
  </sense-list>
</lexeme>
<lexeme id="707" start="4227" length="7">
  <surface-form>harmony</surface-form>
  <sense-list disambiguated="yes">
    <sense id="1">
      <part-of-speech>NOUN</part-of-speech>
      <base-form>harmony</base-form>
    </sense>
  </sense-list>
</lexeme>
```

Figure 4.3: A document tagged by Xelda in XML format

4.4.2 Learning

The `svm_learn` module is used to train the machine and is run at the command line by typing:

```
svm_learn [options] train_file model_file
```

The `train_file` is the input file which was described in section 4.4.1. To recap, this file represents all documents in the training data where each line represents a document and each line consists of `FEATURE:VALUE` pairs. The features represent one of BOW, bigrams or POS tags, depending on which representation is being used at the time.

Default training for the SVM is to use no option. However, various parameters of the SVM can easily be tuned and tweaked during learning and this allows for straight forward cross-validation. Joachims [1998b] points out that “SVMs do not require any parameter tuning, since they can find good parameter settings automatically”. Previous studies have shown the default settings have proven to be the most effective. Despite this, we will alter some of the settings during training to see if this impacts on the effectiveness of the classifier. The parameter settings that can be altered include:

learning options: various learning parameters can be altered. Some of the options include:

- `-c` changing c results in the SVM trading-off between training errors and the margin that separates the data.
- `-j` increasing or decreasing j changes the cost-factor, i.e. adjusts the amount by which training errors of positive examples outweigh negative ones.

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW89

- $-b$ the use of an unbiased hyperplane instead of the default biased one.

performance estimation: to do with estimating the performance of the classifier.

transduction options: relating to the fraction of unlabeled examples to be classified into the positive class.

kernel options: the type of kernel function $-t$ can be altered. The internal parameters of the kernel functions themselves can also be changed.

- linear is the default kernel,
- polynomial - $\langle x, z \rangle^d$,
- radial basis function - $K(x, z) = \exp \frac{-\|xz\|^2}{c} s$,
- sigmoid tanh - $\tanh k_x \cdot y - \delta$.

optimisation options: allows various parameters to be changed in order to optimise kernel evaluations, iterations, or training criteria.

First of all we train the SVM using the default options. Then we examine the effect of tuning the SVM and, in particular, examine the effect of changing the kernel function. We report on the most effective kernel function for each of the BOW, bigrams and POS representations.

When learning is complete on the `train_file`, the result is the learned model which is written to the `model_file`.

4.4.3 Classifying

During classification, the `model_file` produced during SVM learning is used to predict the output class of unseen documents. `svm_classify` is used to classify unseen documents and this is done as follows:

```
svm_classify [options] unseen_file model_file
              output_file
```

The `unseen_file` contains the documents yet to be classified. These documents are represented in the same way that was described in section 4.4.1 and again briefly in section 4.4.2.

The `model_file` is the model which is learned from the training data `train_file` during learning in 4.4.2. The SVM reads this file during classification to make predictions on the output class of each of the documents in `unseen_file` to be classified.

Each line in the `output_file` represents one document in the set of documents to be classified (i.e. `unseen_file`). This line contains the value of the *decision function* (see sections 3.4.2 and 3.4.3 for further information) on a document in `unseen_file`. The sign of this value determines classification, i.e. a positive value indicates the document is racist while a negative value means the document is not racist.

4.5 Evaluation

In chapter 2 we discussed indexing, the first step involved in building the text categorisation system. The second step, classifier construction, was dealt

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW91

with in chapter 3. These two steps are the only ones required in order to build the classification system, but as with any system we cannot know how effective it is until we evaluate it. Evaluation, the last step, is very important for text categorisation and we need empirical evidence to see how good the system is.

As already explained earlier in this chapter, we have split the dataset into a training and test set (see Table 4.2). We will use each test set to evaluate the performance of the classifier.

We use *Precision and Recall* measures to evaluate the system. “Recall and precision are ubiquitous in information retrieval, where they measure the proportion of relevant documents retrieved and the proportion of retrieved documents which are relevant” Lewis [1991]. Precision is the number of categories correctly assigned, divided by the total number of categories assigned. It is a conditional probability defined as $P(ca_{ix} = 1|a_{ix} = 1)$ Sebastiani [2002], i.e. the probability that if a random document d_x is classified under c_i , this decision is correct. Recall is the number of categories correctly assigned, divided by the total number of categories that should be assigned. It is essentially a measure of the degree of coverage for a specific category and is defined as the conditional probability $P(a_{ix} = 1|ca_{ix} = 1)$ (Sebastiani [2002]), i.e. the probability that, if a random document d_x ought to be classified under c_i , this decision is taken.

Accuracy will also be used to evaluate the system. This is calculated on the test set by checking that the suggested tag is the same as the actual tag.

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW92

It is simply calculated as in equation 4.1:

$$(4.1) \quad \frac{\text{Number-of-documents-assigned-to-the-correct-category}}{\text{Number-of-documents-to-be-categorised}}$$

We will also be using the *F1-measure* to evaluate the performance of the classifier. The F1-measure is obtained by first computing precision and recall and then using them to calculate the F1-measure. It “combines precision and recall with equal importance into a single parameter for optimization” (Pierre [2000]). This is a useful measure by which to evaluate the overall performance of the classifier assuming it is of value to treat precision and recall with equal importance. The F1-measure is defined in equation 4.2 as:

$$(4.2) \quad F1 = \frac{2Precision \times Recall}{Precision + Recall}$$

4.6 Summary

In this chapter we took a closer look at the problem of detecting racism online and highlighted how this problem differs from traditional categorisation problems. We described work that has been carried out in University College

CHAPTER 4. TEXT CATEGORISATION FOR RACISM ON THE WWW93

Dublin by Finn and Kushmerick [2002], Finn et al. [2002] and identified the similarities between that research and our own. We presented some of the findings of the PRINCIP project carried out at DCU (see Gibbon and Greevy [2003], Lechleiter and Greevy [2003], Martin [2003a,b]) and explained how this has helped us in forming our methodology for this thesis.

We introduced the methodologies that were used for data collection and presented the statistics on the datasets that were during training and testing of the text categorisation system.

We discussed building the training and test data, as well as the various processes the data must undergo so that the both the training and test data (and indeed any future data to be classified by the machine) are interpretable by the Support Vector Machine. We also discussed the pre-processing stages which the data must undergo in order for it to be turned into the appropriate representations, namely BOW, bigrams or POS tags.

Finally we explained how to train and test the SVM given the data and we introduced the measures used to evaluate the effectiveness of the system.

In the next chapter we will look at the results of the text categorisation system for the identification of racist documents online.

Chapter 5

Results

We took a Support Vector Machine, SVM^{light} , trained it on three representations of the data - BOW, n-grams and POS - and evaluated the effectiveness of each classifier in terms of precision, recall accuracy and F1-measure, as discussed in section 4.5. We looked at the impact on datasets of varying sizes. The effect of tweaking and tuning certain parameters of the SVM is analysed.

Some of the results presented in this chapter were presented in 2004 a paper at the 7th International Conference on the Statistical Analysis of Textual Data in Leuven-la-Neuve, Belgium (Greevy and Smeaton [2004b]) and in a poster at the 2004 27th Annual International ACM SIGIR Conference in Sheffield, UK (Greevy and Smeaton [2004a]).

5.1 Bag of Words

The first experiments we conducted used the BOW representation. In section 4.4.1 we discussed the various processes involved in preparing the data for training.

For these initial experiments we trained the classifier using the default SVM parameter settings. Learning, performance, kernel and optimisation settings remained unchanged. The default kernel is linear. Unless otherwise stated, all results reported are on the linear kernel function. We refer the reader to section 4.4.2 for further information on the SVM parameter settings.

We compared two term weighting measures to analyse the most effective method for weighting terms. We compare the number of occurrences of a term in a document with the frequency of a term in a document (see equation 2.1). In Table 5.1 we see the results of this preliminary experiment. We

Table 5.1: Evaluation of different term weighting measures for BOW

Dataset	Term Weight	Precision	Recall	Accuracy	F1-measure
Set 1	No. occurrences	87.50%	23.33%	60.00%	36.84
Set 1	Frequency	92.31%	80.00%	86.67%	85.72

observe that *frequency* is a more effective measure for term weighting with a F1-measure of 85.72 compared with 36.84 for the number of occurrences as term weight. Since *frequency* as term weight is most effective, we will be using it as means for measuring term weight in the next experiments. Table 5.2 outlines the results of the classifier when BOW is used as a representation. We have tested this representation on datasets of varying sizes. From Table 5.2 and Figure 5.1 we see that, as the the size of the training Set increases, so

Table 5.2: BOW performance

Dataset	Precision	Recall	Accuracy	F1
Set 1	92.31%	80.00%	86.67%	85.72
Set 2	84.00%	84.00%	84.00%	84.00
Set 3	87.84%	86.67%	87.33%	87.25
Set 4	92.55%	87.00%	90.00%	89.69

does recall. High precision is reported for Set 1 and thereafter it takes a slight dip but the best precision and overall result is seen in Set 4. The accuracy and F1 scores behave similarly – dipping after Set 1 and then improving from Set 2 to Set 4. In Figure 5.2 the results are more transparent: the classifier performs best on the largest dataset – Set 4, with a notable improvement in performance from Set 2 to Set 4. Figure 5.2 gives a clearer indication of the

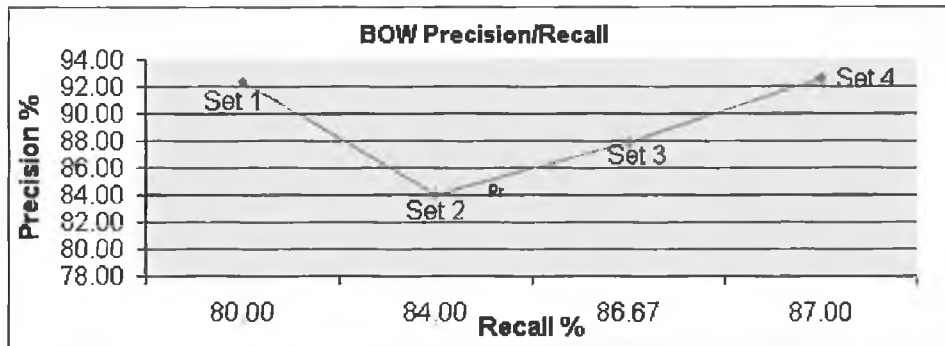


Figure 5.1: BOW: Precision and Recall figures for each dataset

performance of the classifier on the BOW representation for each dataset. Remember that $F1$ treats precision and recall with equal importance. The results in Figure 5.2 are similar to Figure 5.1; the classifier performs best on the largest datasets – Sets 3 and 4.

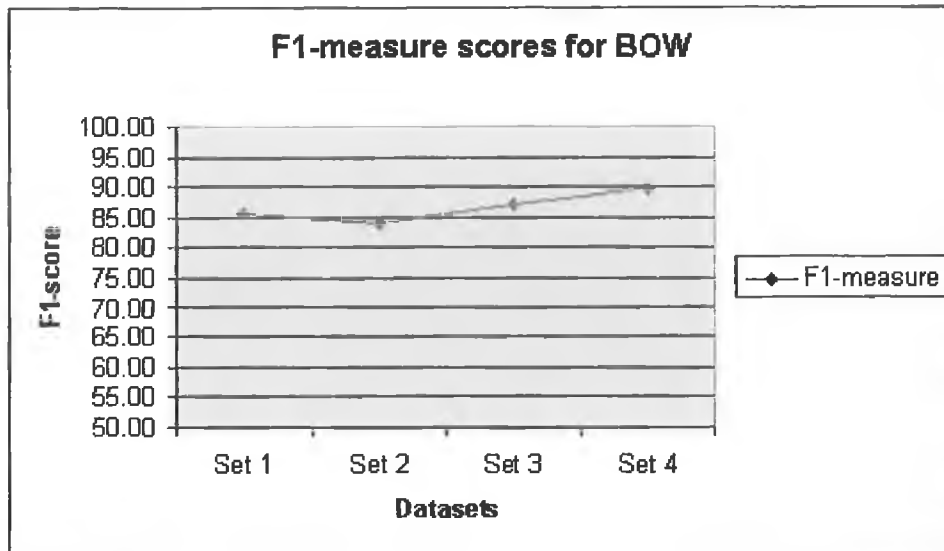


Figure 5.2: BOW: F1-measure scores

5.2 Bigrams

In section 4.4.1 we discussed the steps that must be taken in order to transform the data into an appropriate format for training.

Table 5.3: Comparison of a term weighting measure on BOW and bigrams on Set 1

Representation	Term Weight	Precision	Recall	Accuracy	F1
BOW	No. occurrences	87.50%	23.33%	60.00%	36.84
Bigram	No. occurrences	66.67%	40.00%	60.00%	50.00

In a preliminary experiment (see Table 5.3) using the *number of occurrences* as term weight, we compared the BOW and bigram representations on the smallest dataset, Set 1. From this initial experiment we see that the BOW gives highest precision while the bigrams gives highest recall. The

accuracy on the test set works out the same for each representation but the F1-measure figures differ greatly with the bigrams outperforming the BOW. We will compare and contrast the representations more thoroughly after we look at the performance of the classifier using the bigram representation on the larger datasets.

Table 5.4 reports on the performance of the classifier on each of the datasets. Bigrams are used as a representation in this experiment. From Table 5.4 we see that as the training Set increases precision decreases from

Table 5.4: Bigrams performance

Dataset	Precision	Recall	Accuracy	F1
Set 1	100.00%	48.39%	73.77%	65.22
Set 2	95.00%	74.51%	83.00%	83.52
Set 3	93.44%	75.00%	84.77%	83.21
Set 4	94.12%	80.81%	87.94%	86.96

100% in Set 1 to 94.12% in Set 4, rising slightly between Set 3 and Set 4. Recall on the other hand, increases from as low as 48.39% in Set 1 to as high as 80.81% in Set 4. As the training Set increases, accuracy increases steadily from 73.77% to 87.94%. The F1-measure follows a similar pattern to precision, dipping after Set 2 and rising again between Set 3 and Set 4. This pattern is more obvious in Figure 5.3.

If we compare Figures 5.2 and 5.3, representing the F1 scores for the BOW and bigram representations, we notice that the F1 scores are higher across each dataset for BOW than for bigrams. This is more clearly represented in Figure 5.4.

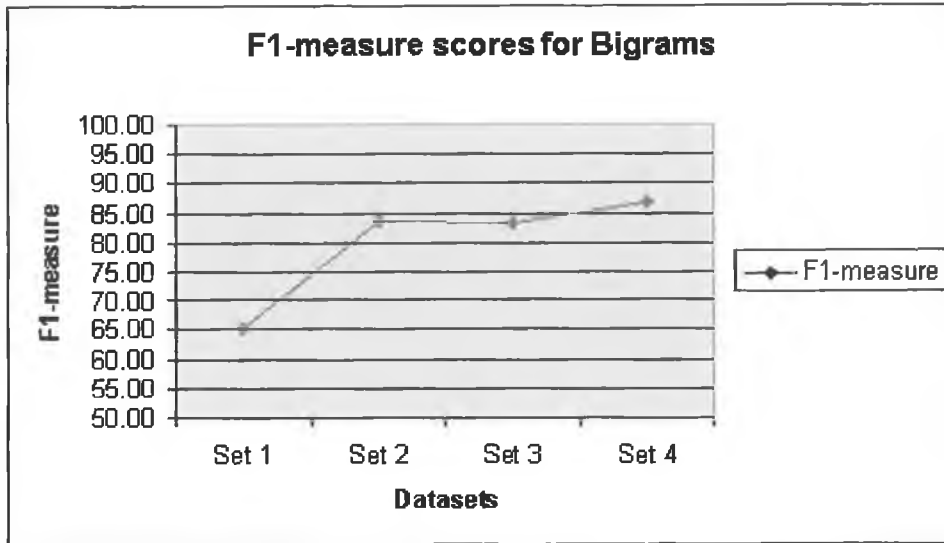


Figure 5.3: Bigrams: F1-measure scores

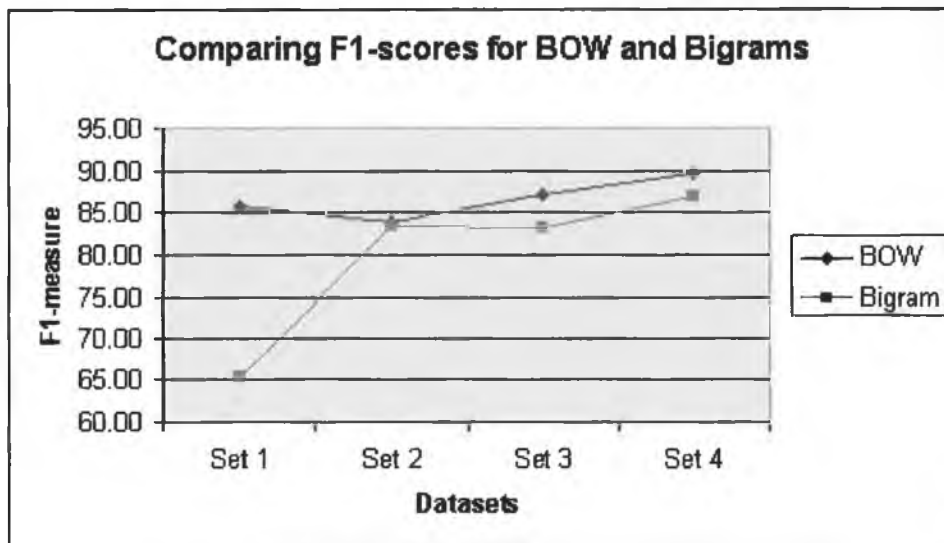


Figure 5.4: Comparing F1 scores for BOW and Bigrams

5.3 Part-of-speech Tags

In section 4.4.1, we discussed the steps that must be taken in order to transform the data into an appropriate format for training.

We trained the classifier using POS as a representation with frequency as term weight. Table 5.5 reports on the performance of the classifier when POS is used as a representation. We see that as the training set size increases precision decreases steadily from 87.10% to 72.58% – i.e. precision is lower

Table 5.5: POS performance

Dataset	Precision	Recall	Accuracy	F1
Set 1	87.10%	90.00%	88.33%	88.53
Set 2	75.81%	94.00%	82.00%	83.93
Set 3	76.67%	92.00%	82.00%	83.64
Set 4	72.58%	90.00%	78.00%	80.36

than what was reported for the BOW and bigram representations. Recall peaked to 94% on Set 2 and then decreases to 90% for Set 4, outperforming both the BOW and bigram representations. Both the accuracy and F1 scores consistently decrease as the training Set increases, meaning overall, POS performs well when the training and test data are small.

5.4 Comparing BOW, Bigrams and POS

In this section we digest and analyse the results which we obtained in our experiments in sections 5.1, 5.2 and 5.3.

5.4.1 Analysis of Performance in terms of the F1-measure

Figure 5.5 illustrates the performance of each of the representations in terms of the F1-measure, an evaluative measure which treats both precision and recall equally. From Figure 5.5 we see that, overall, the BOW representation outperforms the other two. POS performs better than any other representation for Set 1 but we must remember that this is the smallest dataset and so the data on which the classifier is tested is much smaller than the test data in Set 4, the largest of the training data.

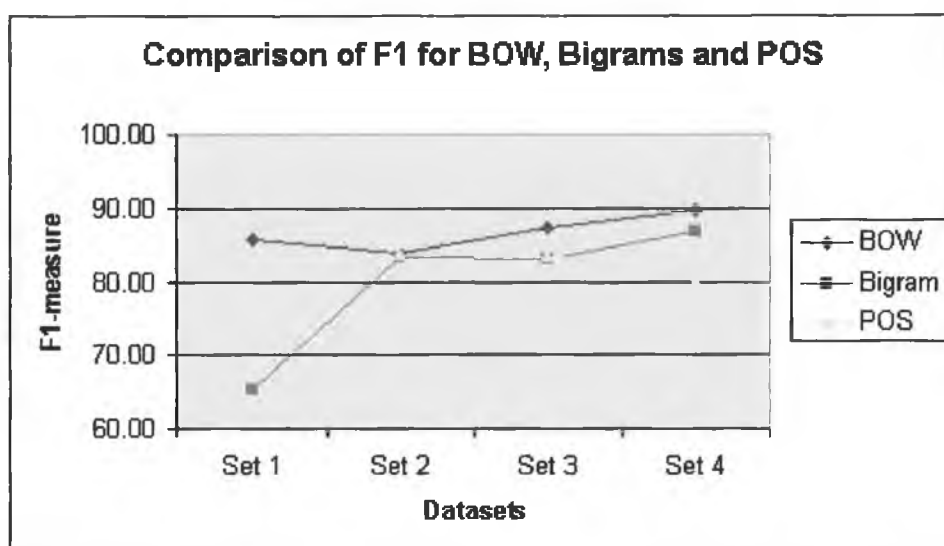


Figure 5.5: Comparison of F1-measure for BOW, Bigrams and POS

5.4.2 Analysis of Performance in terms of Precision

So far we have been using the F1-measure to evaluate the performance of the classifier for each of the representations. Since this measure assumes precision

and recall should be treated equally, the merits of each of the representation are not altogether clear. We want a classifier that assigns documents to correct categories (precision) as well as a classifier with a high degree of coverage (recall). In IR, there is usually some degree of trade-off between precision and recall – low recall is usually sacrificed for high precision. However, the best classification system will have both high recall and high precision and for this reason we will analyse each representation separately in terms of precision and recall. This will allow us to see more clearly the advantages and disadvantages of each of the representations.

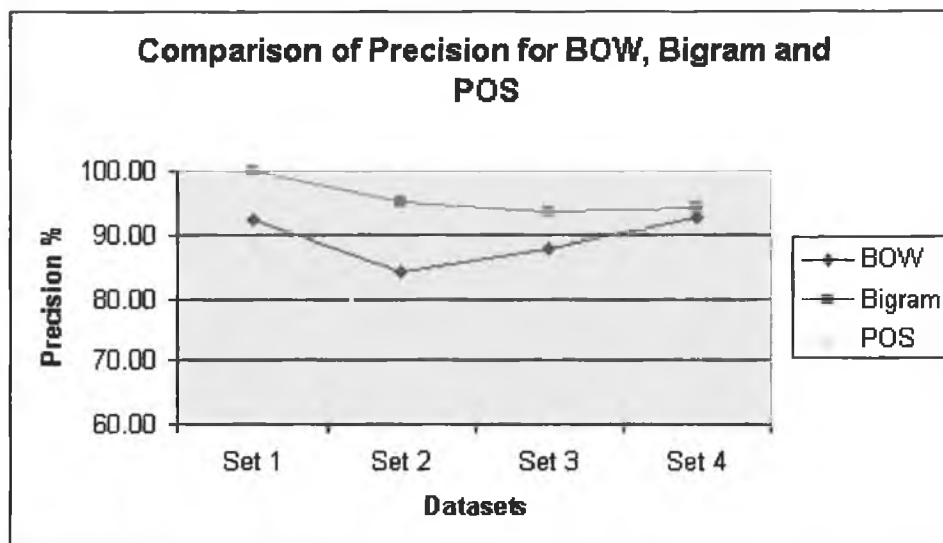


Figure 5.6: Comparison of Precision for BOW, Bigrams and POS

Figure 5.6 illustrates the precision scores for each of BOW, bigrams and POS. Although the bigram representation decreases as the training set increases, it outperforms BOW and POS. BOW precision takes a slight dip for Set 2 but then steadily increases to 92.55%, slightly below bigram precision

at 94.12%.

5.4.3 Analysis of Performance in terms of Recall

Figure 5.7 allows us to easily identify that the POS representation gives highest recall overall. Though recall dips slightly from Set 2 to Set 4 for the

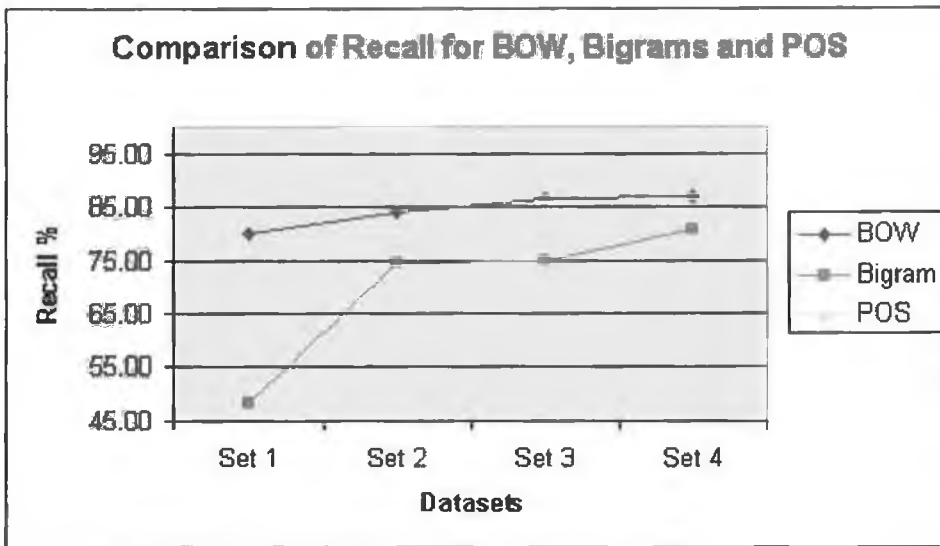


Figure 5.7: Comparison of Recall for BOW, Bigrams and POS

POS representation, it still outperforms the BOW and bigrams. BOW, on the other hand, improves steadily as the training set increases reaching 87%, 3% less than that achieved for the POS representation on Set 4.

5.5 Tuning SVM Parameters

Parameter settings have been experimented with within the SVM and studies have shown that the default settings are the most effective. Joachims [1998b] points out that “SVMs do not require any parameter tuning, since they can find good parameter settings automatically”.

In these next experiments we analyse the effect of changing the kernel options on classification performance. Experiments prior to this section used the linear kernel function. Here, we look at polynomial, sigmoid tanh and the radial basis function as kernel functions.

5.5.1 Polynomial as a Kernel Function

BOW

Table 5.6 reports on the performance of the classifier on the BOW representation when the polynomial kernel function is used. The results follow a similar pattern to those reported in the previous section in Table 5.2 where the linear kernel function is used. However, the performance improved slightly for the polynomial kernel – though the results achieved on Set 1 are the same for

Table 5.6: BOW performance using the polynomial kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	92.31%	80.00%	86.67%	85.72
Set 2	85.71%	84.00%	85.00%	84.85
Set 3	87.84%	86.67%	87.33%	87.25
Set 4	92.78%	90.00%	91.50%	91.37

both the polynomial and linear kernel functions, improved precision, recall,

accuracy and F1 scores were recorded for Sets 2, 3 and 4.

Bigrams

Table 5.7 presents the performance scores for the bigram representation using polynomial as a kernel function. When we compare this table to Table 5.4, where the linear kernel function is used, we see that precision deteriorates significantly for Set 1. We see an increase for Sets 2 and 3 to the same as was reported for the linear kernel function in Table 5.4, with a slight improvement

Table 5.7: Bigram performance using the polynomial kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	50.82%	100.00%	50.82%	67.39
Set 2	95.00%	74.51%	85.15%	83.52
Set 3	93.44%	75.00%	84.77%	83.21
Set 4	94.20%	65.66%	80.90%	77.38

in precision for Set 4. Recall improves significantly for Set 1. Again exactly the same figures are reported for Sets 2 and 3 as were reported in Table 5.4. Recall dipped again for Set 4. This dramatic difference between precision and recall for Set 1 when the linear and polynomial kernels are used may be due to the small size of the dataset. Overall, these figures lead to a higher F1 score for Set 4, lower for Set 1 and the same for Sets 2 and 3. Overall, the polynomial function does not improve the performance of the classifier for the bigram representation.

POS

Table 5.8 shows the results of the polynomial kernel function on the POS representation. We compare these results with those obtained for the linear function in Table 5.5 and note an improvement in precision for all datasets.

Table 5.8: POS performance using the polynomial kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	93.10%	90.00%	91.67%	91.52
Set 2	75.81%	94.00%	82.00%	83.93
Set 3	77.53%	92.00%	82.67%	84.15
Set 4	73.17%	90.00%	78.50%	80.72

Recall remains the same. Both accuracy and the F1-measure improve because of the improved precision. Therefore, we conclude that the polynomial kernel function outperforms the linear for the POS representation.

Comparing BOW, Bigrams, POS

In Figure 5.8 we see a similar pattern to that in Figure 5.5. Although POS performs best for Set 1 (as we also saw in Figure 5.5), BOW outperforms both bigrams and POS for Sets 2 to 4. The F1-measure treats precision and recall with equal importance and so this evaluative measure tells us that overall the POS representation performs better than both the bigrams and POS.

In Figure 5.9 we see a pattern not dissimilar to that in Figure 5.6 – however Set 1 suffers greatly, with a significant drop in precision. This could however be due to the rather small number of documents in the dataset. Precision quickly increases thereafter and outperforms the BOW and POS

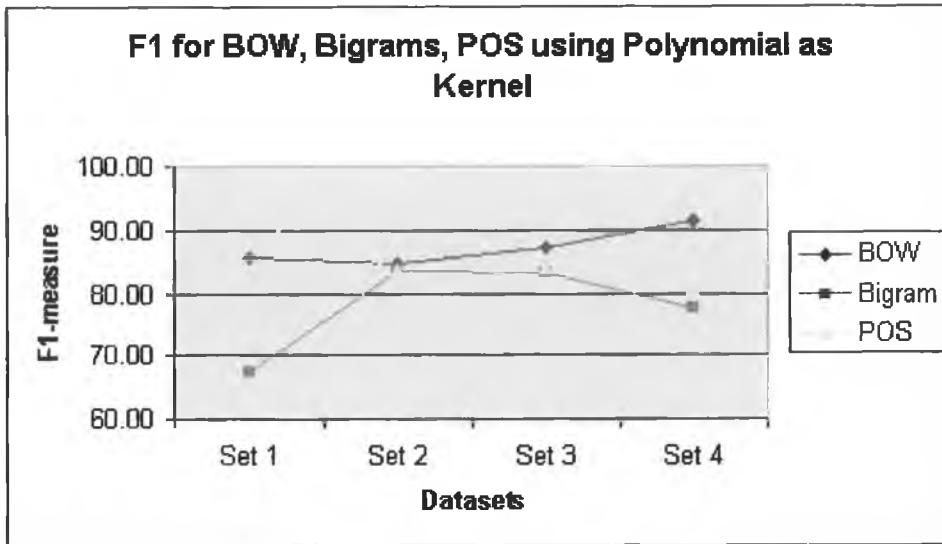


Figure 5.8: Comparison of F1-measure for BOW, Bigrams and POS using Polynomial Kernel Function

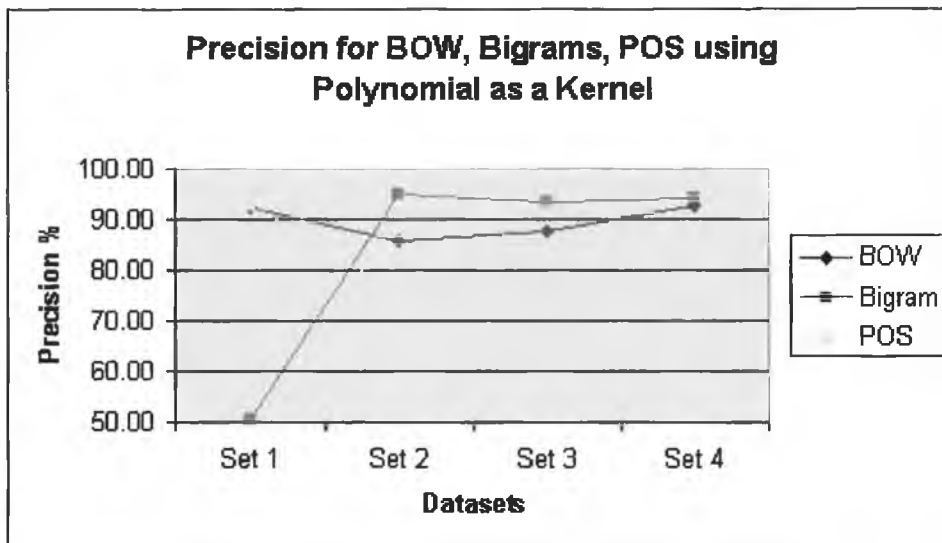


Figure 5.9: Comparison of Precision for BOW, Bigrams and POS using Polynomial Kernel Function

representation for Sets 2 to 4. Precision remains the same as the linear for Sets 2 and 3 and slightly rises for Set 4. We conclude again that of the three representations, bigrams leads to the best precision, though the figures were more consistent for the linear function.

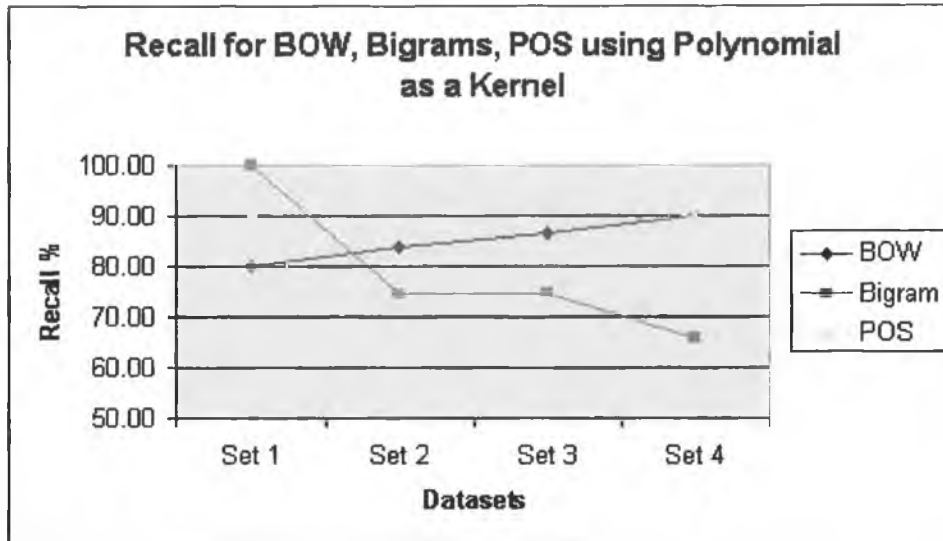


Figure 5.10: Comparison of Recall for BOW, Bigrams and POS using Polynomial Kernel Function

In Figure 5.10, again we see a pattern not unlike that in Figure 5.7. Bigrams perform best for Set 1 but POS then outperforms BOW and bigrams for Sets 2, 3 and 4. The recall figures for POS remain the same for the polynomial as those reported for the linear kernel function in Figure 5.7.

5.5.2 Sigmoid tanh as a Kernel Function

BOW

Table 5.9 represents the results of the BOW representation when sigmoid tanh is used as a kernel function. The results follow a similar trend to those obtained for both the linear and polynomial kernel functions. However, the scores diminish slightly compared to those obtained for the linear kernel function – precision drops by between about .33% and 1% for Sets 2 and 4 and recall drops by 2% for Set 2. The other scores remain the same. The polynomial kernel function outperforms sigmoid tanh as well. Therefore, we

Table 5.9: BOW performance using the sigmoid tanh kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	92.31%	80.00%	86.67%	85.72
Set 2	83.67%	82.00%	83.00%	82.83
Set 3	87.84%	86.67%	87.33%	87.25
Set 4	91.58%	87.00%	89.50%	89.23

conclude that the sigmoid tanh kernel function does not result in improved performance.

Bigrams

Table 5.10 displays the results of the bigram representation when sigmoid tanh is used as a kernel function. When we compare the results with the linear kernel, we see that the sigmoid tanh kernel function improves precision for Set 4 and improves recall for Sets 1 and 4. Compared to the polynomial, sigmoid tanh results in improved precision for Sets 1 and 4, while recall takes

a dip for Set 1, it improves significantly for Set 4. As already mentioned, the dramatic difference in performance for Set 1 may be due to size. Overall, we

Table 5.10: Bigram performance using the sigmoid tanh kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	100.00%	67.74%	83.61%	80.77
Set 2	95.00%	74.51%	85.15%	83.52
Set 3	93.44%	75.00%	84.77%	83.21
Set 4	94.32%	83.84%	89.45%	88.77

see improved accuracy and F1-measures and can therefore conclude that the sigmoid tanh kernel function performs better than both the polynomial and linear kernel functions.

POS

From Table 5.11 we see the results of the sigmoid tanh kernel function on the POS representation. Performance diminishes compared to the results obtained for both the linear and polynomial kernel functions. Precision on Sets 3 and 4 drop and recall on Set 4 drops compared to those achieved using the linear kernel. When compared to the polynomial we also see a drop in

Table 5.11: POS performance using the sigmoid tanh kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	87.10%	90.00%	88.33%	88.53
Set 2	75.81%	94.00%	82.00%	83.93
Set 3	77.53%	92.00%	82.67%	84.15
Set 4	72.36%	89.00%	77.50%	79.82

precision for Sets 1 and 4 and in recall for Set 4. This in turn leads to a

decline in accuracy and F1-measures. The sigmoid tanh kernel function does not improve the performance of the classifier for the POS representation.

Comparing BOW, Bigrams, POS

From Figure 5.11 we see that overall the BOW outperforms the other representations. Similar findings were reported in Figures 5.5 and 5.8 where it was

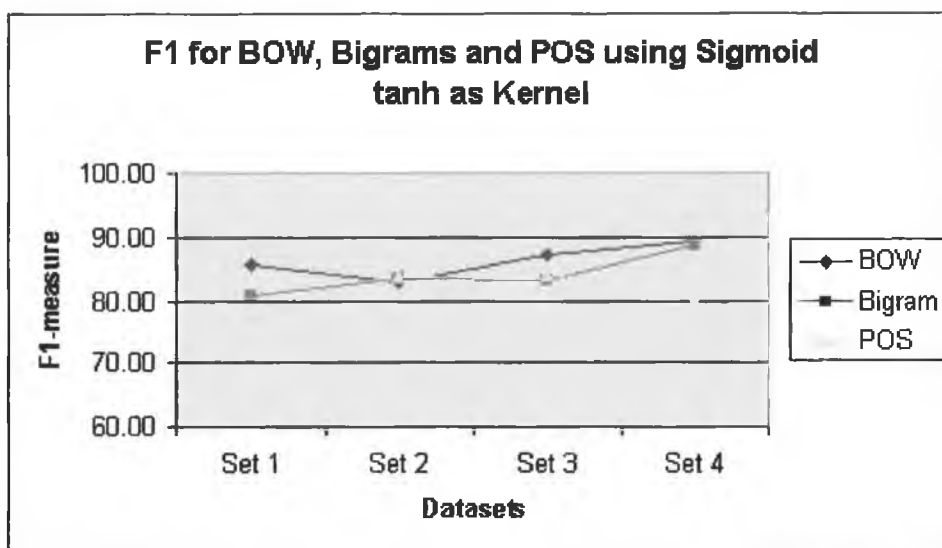


Figure 5.11: Comparison of F1 for BOW, Bigrams and POS using Sigmoid tanh Kernel Function

found that POS performed best for Set 1, all three representations performed equally well for Set 2 and from there the BOW improved, outperforming the bigrams and POS by between 5 – 10%.

Figure 5.12 shows the precision figures for each of BOW, bigram and POS representations, using the sigmoid tanh as a kernel function. In Figure 5.12 we see that bigrams give the highest precision, and BOW and POS gives the

lowest. The same pattern is reported when the linear and polynomial kernel functions are used. For bigrams, precision declines from Set 1 to 3 and then rises for Set 4 for both the linear and sigmoid tanh kernels with sigmoid tanh

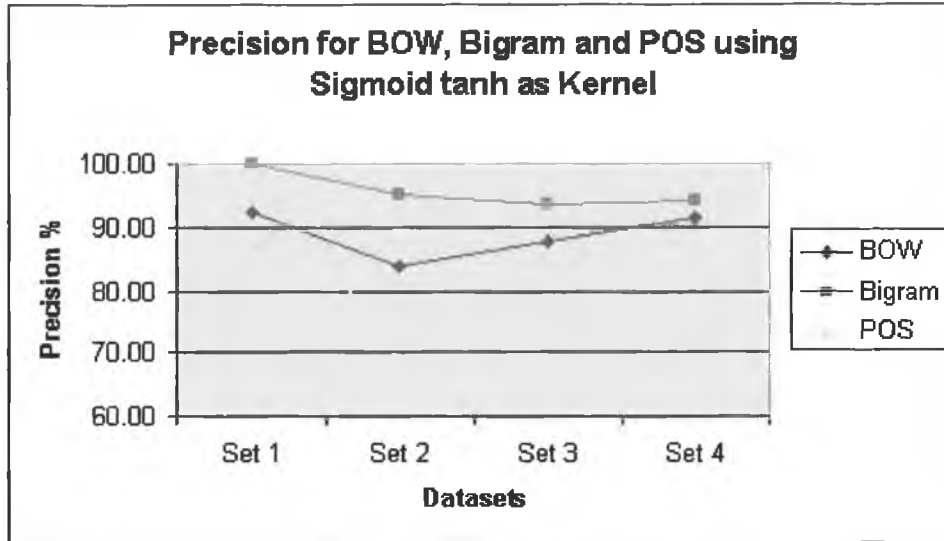


Figure 5.12: Comparison of Precision for BOW, Bigrams and POS using Sigmoid tanh Kernel Function

giving slightly higher precision for Set 4. Our findings thus far show that bigrams give the highest precision.

Figure 5.13 shows the recall figures for each of BOW, bigram and POS representation using the sigmoid tanh as a kernel function. In Figure 5.13 we see that the POS representation gives a higher recall than both the BOW and POS representations, outperforming both by between about 5 – 23% for Sets 1 to 3. BOW recall improves greatly for Set 4 reaching 87% but does not outperform POS at 89%.

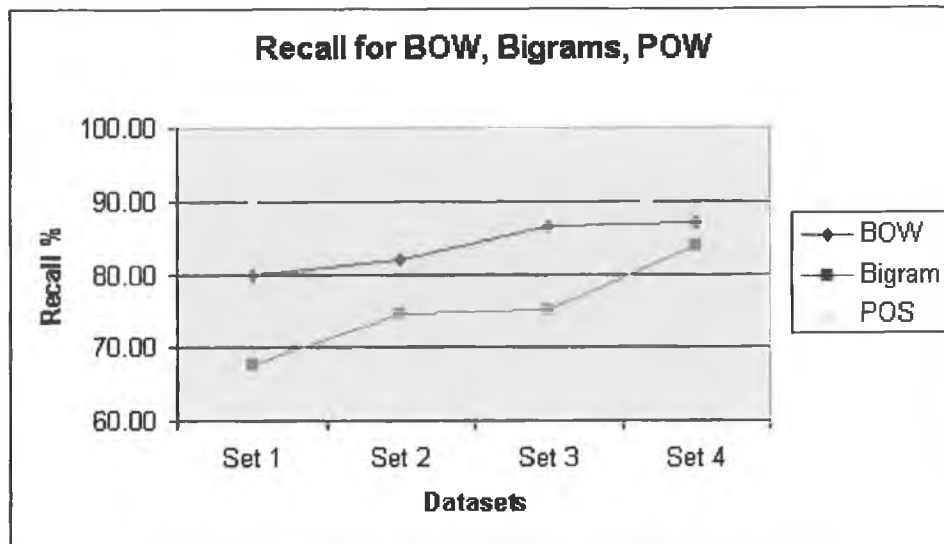


Figure 5.13: Comparison of Recall for BOW, Bigrams and POS using Sigmoid tanh Kernel Function

5.5.3 Radial Basis Function as a Kernel Function

BOW

Table 5.12 displays the results of using the radial basis function (RBF) as a kernel function on the BOW representation. By comparing Table 5.12 with Tables 5.2, 5.6 and 5.9 we see that although the RBF kernel gives better

Table 5.12: BOW performance using the radial basis function as a kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	92.31%	80.00%	86.67%	85.72
Set 2	85.71%	84.00%	85.00%	84.85
Set 3	86.67%	86.67%	86.67%	86.67
Set 4	92.78%	90.00%	91.50%	91.37

results for precision than the linear and sigmoid tanh kernels, it does not perform as well as the polynomial. The recall scores for RBF are on a par with the polynomial figures. Therefore, we conclude that polynomial gives the best performance for the BOW representation.

Bigrams

Table 5.13 displays the results of the RBF on the bigram representation. By comparing Table 5.13 with Tables 5.4, 5.7 and 5.10 we see that the precision scores are the same as sigmoid tanh and linear for Sets 1 and 2, and less for Sets 3 and 4. Recall is the same as linear, polynomial and sigmoid tanh for

Table 5.13: Bigram performance using the radial basis function as a kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	100.00%	67.74%	83.61%	80.77
Set 2	95.00%	74.51%	85.15%	83.52
Set 3	91.94%	75.00%	84.11%	82.61
Set 4	93.10%	81.82%	87.94%	87.10

Sets 2 and 3. Although it is better than the linear and polynomial for Set 4, it does not reach the performance of sigmoid tanh – 83.84% on Set 4. Overall the sigmoid tanh produces the best F1 scores for the bigram representation.

POS

Table 5.14 represents the results of using the RBF as a kernel function on the POS representation. When we compare Table 5.14 to Tables 5.5, 5.8 and 5.11, we see that the precision scores on Sets 1 to 3 for RBF are equal to those obtained using the polynomial kernel function. Precision on Set 4 drops

Table 5.14: POS performance using the radial basis function as a kernel

Dataset	Precision	Recall	Accuracy	F1
Set 1	93.10%	90.00%	91.67%	91.52
Set 2	75.81%	94.00%	82.00%	83.93
Set 3	77.53%	92.00%	82.67%	84.15
Set 4	72.58%	90.00%	78.00%	80.36

to 72.58% with the polynomial producing the best precision (73.17%). The recall for Sets 1-4 are the same for the linear, polynomial and RBF kernel functions. Overall, the polynomial kernel function gives the best performance for the POS representation.

Comparing BOW, Bigrams and POS

Figure 5.14 illustrates a comparison of the F1 scores for the BOW, bigram and POS representations on each dataset when the RBF is used as a kernel function. We see a similar pattern to Figures 5.5, 5.8 and 5.11 in that overall the BOW proves to be the most effective representation for the classification of racist texts.

The next two figures take a closer look at the strong points of each of the representations by examining which give the highest precision and recall. Figure 5.15 represents the precision scores of each representation on each dataset using the RBF as a kernel function. Again, as in Figures 5.6, 5.9 and 5.12, we see the bigram representation gives the highest precision, though for Set 4 the BOW is just .33% behind the bigrams.

Figure 5.16 represents the recall scores of the each representations for each dataset using the RBF as a kernel function. As in Figures 5.7, 5.10 and 5.13,

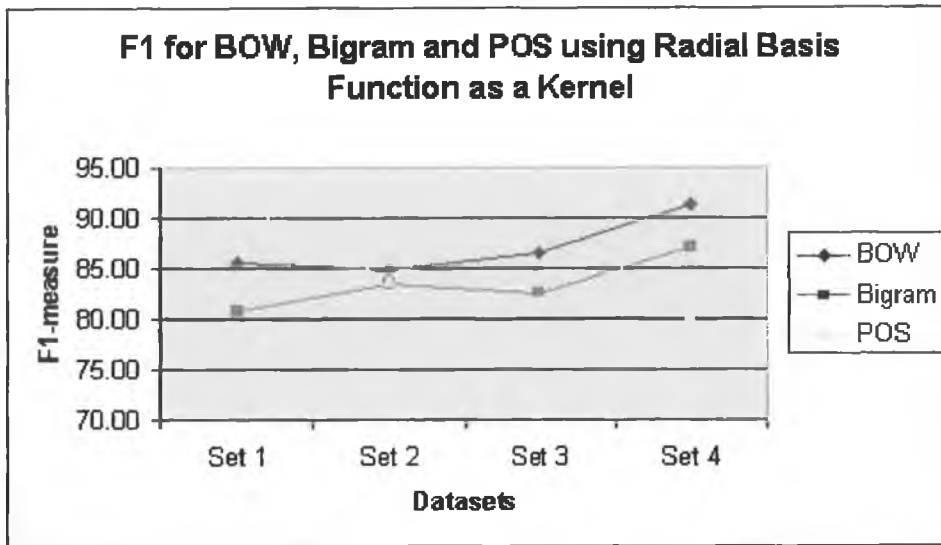


Figure 5.14: Comparison of F1 for BOW, Bigrams and POS using Radial Basis Function as a Kernel

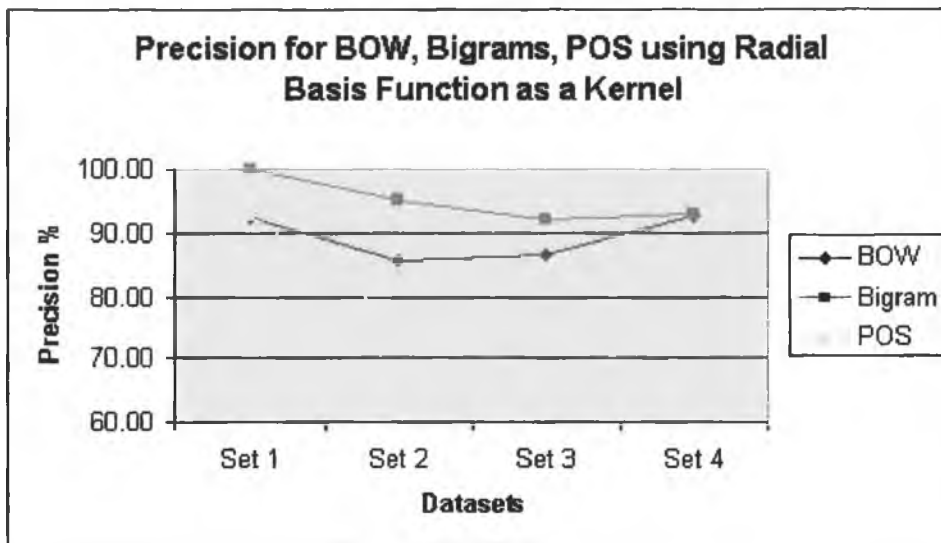


Figure 5.15: Comparison of Precision for BOW, Bigrams and POS using Radial Basis Function as a Kernel

we see that the POS representation gives the highest recall overall, though for Set 4 the BOW and bigrams are level with a recall of 90%. POS performance declines from Set 2 to Set 4 whereas BOW recall increases consistently from Set 1 to 4.

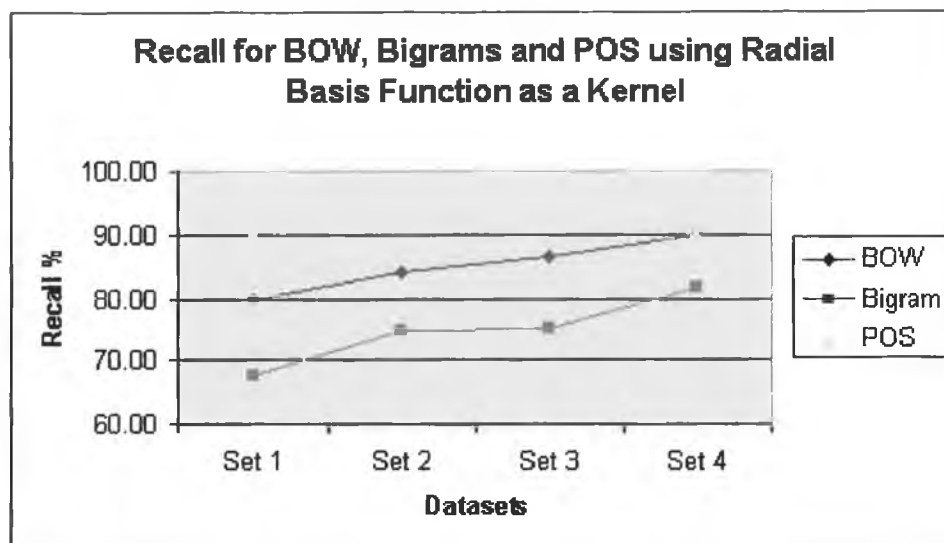


Figure 5.16: Comparison of Recall for BOW, Bigrams and POS using Radial Basis Function as a Kernel

5.6 Summary

In this chapter we looked at three representations of the data within the SVM – bag-of-words, bigrams and part-of-speech. We have analysed each of these representations in terms of precision, recall, accuracy and F1-measure, identifying the pros and cons of each approach and the most effective representation overall. We also analysed the effect of using different kernel functions within

the SVM – linear, polynomial, sigmoid tanh and radial basis function. Each kernel was tested on each representation, thereby enabling us to identify the most effective kernel for each representation. The results obtained in this chapter will be scrutinised and analysed in more detail in the next chapter.

Chapter 6

Conclusion

6.1 Overview

Since the advent of the Internet in the early 90s and the consequent introduction of HTML, there has been a surge in the availability of documents in electronic format. Information which was previously published in newspapers, magazines, billboards, journals, newsletters, flyers or as graffiti on walls is now easily distributed on the Internet. HTML is a language used for publishing documents on the Internet and the relative ease with which it can be learned and used makes it accessible to people from all spheres of life and all technical competencies. The nature of the Internet means that documents can be posted anonymously, allowing authors with politically incorrect views or opinions largely unfavourable to the general public to remain, to a large extent, unknown to the general user of the Internet. In the past, such people, being largely a minority, would have found it difficult to express their views publicly and freely and would also have had difficulty in meeting others with

similar viewpoints. The Internet now presents them with a means of interaction, and has enabled such people to congregate and form communities online and to organise themselves to a greater extent. The nature of the Internet as a relatively anonymous and globally available medium, makes it a very attractive means of distributing information. For these reasons, it is a valuable asset to groups whose activities have remained largely underground until now.

In this thesis we have looked at the application of automatic text categorisation for the problem of racism on the Internet. Racism on the Internet is widespread, but especially prominent in the USA due to the First Amendment - the right to freedom of speech. Common targets of racism include Jewish people, people of African descent, refugees and immigrants. However, since September 11th we have seen more racism towards Muslims and Arabs in Western countries and a growth in racism towards white people from countries like the UK and USA. World events, politics, war and current affairs, all influence the targets of racism, which in turn influence content on the Internet. The Internet is growing all the time with websites being edited or added every day. The dataset used in this thesis was collected between February and September 2002 and recent investigations have shown it to be already out of date. Automatic techniques have proven successful for a wide range of classification problems such as news story categorisation, categorisation of documents into Yahoo!-like directory structures or categorisation of classified adverts. Automatic methods make it possible for classifiers to be updated with relative ease in comparison to the manual approach to classification. Given the demands in consistently protecting the younger, more

impressionable users of a constantly changing Internet, it is a classification problem that would benefit from automatic categorisation techniques.

Detecting racism is unlike other topic-based problems (e.g. news story categorisation) that have been successfully dealt with using automatic techniques. If we were to filter news stories into categories such as sport, politics or finance, we might imagine that words such as `football`, `score`, `aggregate`, `foul` and `off-side` would be typical of sports while `ISEQ`, `trading`, `merger` and `index`, on the other hand, would be associated with finance. Detecting racism is not solely dependent on the presence or absence of key words or terms, as such words can appear in potentially any text type – racist, anti-racist, political, or otherwise. We again refer the reader to Figures 4.1 and 4.2 for an example of a racist and a non-racist text which contain the same words. In our search for racist documents during the corpus collection phase of the PRINCIP project, we encountered documents about breeding horses through the search terms `overbreeding + race` and the webpage of a limousine service using the search terms `white knight`¹ – so collocations or co-occurrences of what seem to be racist terms do not always yield racist webpages.

To date, the problem of detecting racism has not been dealt with using automatic text categorisation methods. With the exception of the PRINCIP project,² classification has been mostly manual, with offensive pages being labelled as such and then being added to a block list for use in filtering software. In this thesis we have used Support Vector Machines for the construction of a classifier to handle the categorisation of racism on the net.

¹Note: white knights is a term used to refer to the Ku Klux Klan in the USA

²<http://www.princip.net>

In section 4.3.1 we reported on how we gathered the racist and non-racist datasets. Any kind of automatic classifier needs a training set, which in our case is used to train the machine to identify racism. We are faced with a choice of how to represent each page to be classified. In this concluding chapter, we look at each representation used within the Support Vector Machine. We summarise and analyse the results obtained using these representations (cf. ?? for further information on building the representations), and we highlight the advantages and disadvantages of each representation. We look at the machine learning method used and explain why we used this method above others. We also identify future experiments that should be considered in this area.

6.2 Which Representation?

From the experiments conducted and reported in the last chapter we see that the bigram representation gives the highest precision and POS gives the highest recall. Using the F1-measure to evaluate the system, it is clear from Tables 5.5, 5.8, 5.11 and 5.14 that the BOW representation is the most effective of the three.

For each of the kernel functions tested, we noticed a similar pattern between the performance of the BOW and bigram representations – precision dropped between Sets 1 and 2 for BOW and Sets 1 to 3 for bigrams and then rose again from Set 2 to 4 for BOW and Sets 3 to 4 for bigrams. A larger dataset might actually see the BOW outperforming the bigram representation.

When we used the polynomial as a kernel function, the same recall was reported for the BOW and POS representations on Set 4, the largest dataset. In our experiments, we found that as the dataset increased, recall consistently reduced for the POS representation, whereas it increased consistently for the BOW representation (see Tables 5.7, 5.10, 5.13 and 5.16). It would be interesting to see how each representation would perform on a larger dataset and if the BOW would in fact outperform the POS.

From these experiments, we learned that the BOW is the most effective representation, giving us precision and recall scores almost on a par with the high scores that the bigrams achieved for precision and the POS achieved for recall.

6.3 Which Classification Tool?

There are many machine learning methods that can be used to build a classifier – Neural Networks, Naïve Bayes, Decision Trees, Rocchio, etc. These methods were introduced in chapter 3. For reasons outlined in section 3.3.10, we decided to use Support Vector Machines, a machine learning method introduced by Vapnik Vapnik [1995] and implemented by Joachims in *SVM^{light}*. SVMs are a newer learning method that “since its introduction has already outperformed most other systems in a wide variety of applications” Cristianini and Shawe-Taylor [2000]. SVMs are capable of overcoming many issues which pose problems for other machine learning methods – they are efficient even when dealing with very large datasets with many thousands of features. Therefore, feature selection and extraction do not have to be applied,

as SVMs are capable of finding good solutions in high dimensional spaces. Evaluation is also efficient, making the classifier fast at classifying unseen input.

In training, parameters inside the SVM are easily accessible, meaning the internal structure of the SVM and how the SVM learns can be tweaked to find the optimal solution for the problem at hand. In our work, we analysed the impact of changing the kernel function (we refer the reader to section 3.4.5 for further information on kernel functions) – see section 6.4 below for a summary of the impact of experimenting with kernels.

6.4 Which Kernel?

We evaluated each of the kernel functions in the SVM and of the linear, polynomial, sigmoid tanh and radial basis function, we found the polynomial to give the best scores for the BOW representation, the sigmoid tanh proved most effective for the bigram representation while the polynomial resulted in the best performance for the POS representation.

Using the polynomial as a kernel function resulted in the same recall being achieved on Set 4 for the BOW and POS representations.

We learned that the default linear kernel did not prove to be the most effective for this classification problem. Polynomial performed best for BOW and POS and sigmoid tanh proved best for bigrams.

6.5 Future Work, Criticisms and Conclusions

We have shown that it is possible to construct an automatic text categorisation system capable of detecting racism on the web. We have shown that the BOW representation has proven to be the most effective representation and the best performance is achieved when the polynomial kernel function is used. The BOW approach has similarly proven to be the most effective and efficient representation for many classification problems (Smeaton [1997] and Lewis [1992]). Until now, TC techniques have been applied largely to topic-based problems such as news story categorisation – problems which are typically reliant on keywords to separate classes. So it is surprising to see that the same representation which works well for topic-based problems also appears to work best for the detection of racism, a classification problem which is largely related to attitude or opinion detection, something which is orthogonal to the topic. Finn et al. [2002] and Finn and Kushmerick [2002] reported similar findings for subjectivity classification – a problem also related to opinion. Like us, they found the BOWs approach to perform best. Our results seem a little unusual, as we have shown that racist words and phrases can appear in racist and non-racist texts and, from the outset, we envisaged that an approach based on keywords alone would not be enough to discriminate between racist and non-racist texts. However, our results suggest otherwise. While the BOW performed best overall, the bigram representation resulted in the best precision. This is to be expected, as the bigram representation conveys context and some notion of sense. Though the same words very often appear in both racist and non-racist texts, the company they keep, or the context in which they are appear has to be radi-

cally different for a text to be considered racist. For instance, our search using `overbreeding + race` retrieved texts on breeding horses as well as texts on African-Americans overbreeding in the suburbs of big cities in the USA – a typical bigram in the former document might be `overbreeding horses`, while in the latter we might expect to find `overbreeding niggers`. Both bigrams are very discriminating and would be far more accurate than just `overbreeding` and `niggers`. Though the bigram representation resulted in the best precision, recall suffered and overall the BOW proved most effective.

There is a lot of scope for future work in this area. We have used just one machine learning method – Support Vector Machines – to build the classifier because it “has already outperformed most other systems in a wide variety of applications” Cristianini and Shawe-Taylor [2000]. However, many of the other methods that we talked about in chapter 3 could be tried for this problem. The Naïve Bayes probabilistic method, which is renowned for its ease of implementation and surprising effectiveness would certainly be one to try, in spite of its simplicity. However, as with the other machine learning methods, the Naïve Bayes requires some element of dimensionality reduction in order to reduce the size of the feature set, otherwise overfitting would cause problems. SVMs on the other hand are capable of coping with high dimensional feature spaces and therefore overfitting does not impose on learning effectiveness, thereby making it the most appropriate choice of classifier.

In building any classification system, evaluation is of the utmost importance. The system must be tested thoroughly and extensively. Unfortunately, we were restricted to a limited dataset that was constructed between

February and September 2002. The targets of racism have already changed since that time due to world events – for instance the war in Iraq. Further experiments on the current web would tell us more about the classification system.

This system was built without performing stop-word removal or stemming. These are typically performed in order to remove redundant information, reduce variance among features (for instance to have **rac** in place of **race**, **aces**, **racist**, **racism**, **racial**, **racists**), to reduce the size of the feature set so as to avoid overfitting and make the classifier computationally inexpensive and more efficient. Large feature sets can lead to overfitting which impacts on the performance of the classifier. However, SVMs are capable of dealing with many features and capable of generalising well in high dimensional spaces – so it is not necessary to perform stop-word removal or stemming . Also, because of the nature of racism and the use of features of language that are present in potentially any discourse, we avoided such pre-processing tasks in these experiments. Nonetheless, interesting future work would be to see if such pre-processing steps would have an impact on the classification system. Lemmatisation has also proven to “yield a significant improvement in recall” in some experiments Kraaij and Pohlmann [1996] – since the bigram representation proved to give the highest recall overall, it would be interesting to see if the use of lemmas in this representation would influence recall.

In our experiments we compared the number of occurrences and frequency as term weights and found frequency to perform best. Further experiments could be conducted using more sophisticated term weighting measures, such

as TF*IDF.

During the PRINCIP project we identified what we call ‘rules’, which are essentially features that characterise web-based racist discourse. These rules were collected through manual linguistic investigations of the datasets and are fed to the PRINCIP system to allow for the detection of racist webpages. Another interesting study would be to use these rules as features in the SVM in place of the BOW, bigrams or POS representations, thereby allowing us to analyse a combinatory approach – the use of handmade linguistic rules with a machine learning method.

There are many ways that POS tags can be used in categorisation. One relevant paper in the literature was that of Musuyama and Nakagawa [2004] who tried to improve the performance of classifiers using POS tags by extracting combinations of POS tags – for instance all nouns, verbs, adjectives and adverbs were extracted and these words were used to train the classifier. Since these parts-of-speech are largely content words and contain most of the information, they may prove useful in discriminating between racist and non-racist texts.

From chapter 5 we learned that POS achieved the best recall and bigrams achieved the best precision. One obvious experiment would be to combine POS and bigrams to see if we can achieve both high recall and high precision. As previously mentioned BOW achieved the same recall on Set 4 as bigrams when the polynomial kernel function was used – so it would also be interesting to combine POS and BOW.

Bibliography

- C. Apté, F. Damerau, and S.M. Weiss. Automated learning of decision rules for text categorization. *Information Systems*, 12(3):233–251, 1994.
- C.J.C. Burges. Simplified support vector decisions. In *the Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, Bari, Italy, 1996.
- C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- K.M.A. Chai, H.T. Ng, and H.L. Chieu. Bayesian online classifiers for text classification and filtering. In *the Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 97–104, Tampere, Finland, 2002.
- R. Chandrasekar and B. Srinivas. Using syntactic information in document filtering: A comparative study of part-of-speech tagging and supertagging. In *the Proceedings of the 5th RIAO Conference on Computer-Assisted Information Searching on the Internet*, Montreal, Canada, 1997.
- W.W. Cohen. Fast effective rule induction. In *the Proceedings of the 12th In-*

- ternational Conference on Machine Learning*, pages 115–123, Tahoe City, California, USA, 1995.
- W.W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In *the Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–315, Zürich, Switzerland, 1996.
- C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.
- N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.
- H. de Kroon, T. Mitchell, and E. Kerckhoffs. Improving learning accuracy in information filtering. In *the Proceedings of the 13th International Conference on Machine Learning, Workshop on Machine Learning Meets HCI*, Bari, Italy, 1996.
- S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- T. Van Dijk. *Communicating Racism. Ethnic Prejudice in Thought and Talk*. Newbury Park, CA Sage, 1987.
- H. Drucker, B. Shoham, and D.C. Gibbon. Relevance feedback using support vector machines. In *the Proceedings of the 18th International Conference on Machine Learning*, The Berkshires, Massachusetts, USA, 2001.

- S.T. Dumais, D.D. Lewis, and F. Sebastiani. Report on the workshop on operational text classification systems. In *the Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, 2002.
- S.T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *the Proceedings of the 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Washington, USA, 1998.
- J. Allan (Editor), J. Aslam, N. Belkin, C. Buckley, J. Callan, B. Croft (Editor), S. Dumais, N. Fuhr, D.J. Harper, D. Hiemstra, W. Kraaij, D. Harman, E. Hovy, D. Lewis, T. Hofmann, J. Lafferty, V. Lavrenko, L. Liddy, A. McCallum, R. Manmatha, J. Ponte, J. Prager, D. Radev, P. Resnik, S. Robertson, R. Rosenfeld, S. Roukos, M. Sanderson, R. Schwartz, A. Singhal, A. Smeaton, H. Turtle, R. Weischedel, E. Voorhees, J. Xu, and C. Zhai. Challenges in information retrieval and language modelling. In *Report of a Workshop held at the Center for Intelligent Information Retrieval*, University of Massachusetts Amherst, USA, 2002.
- ePrivacy Group. <http://www.eprivacygroup.com/pdfs/spambythenumbers.pdf>, 2003. Last visited 30th April 2004.
- A. Finn and N. Kushmerick. Learning to classify documents according to genre. In *IJCAI-2003 Workshop on Computational Approaches to Text Style and Synthesis*, Acapulco, Mexico, 2002.
- A. Finn, N. Kushmerick, and B. Smyth. Genre classification and domain

- transfer for information filtering. In *the Proceedings of the European Colloquium on Information Retrieval Research*, Glasgow, Scotland, 2002.
- D. Freitag. Using grammatical inference to improve precision in information extraction. In *Workshop on Grammatical Inference, Automata Induction, and Language Acquisition, in the Proceedings of the 14th International Conference on Machine Learning*, Nashville, Tennessee, USA, 1997.
- F. Fukumoto and Y. Suzuki. Manipulating large corpora for text categorization. In *the Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–203, University of Pennsylvania, Philadelphia, USA, 2002.
- J. Fürnkranz. A study using n-gram features for text categorization. *Technical Report OEFAI-TR-9830, Austrian Institute for Artificial Intelligence*, 1998.
- J. Fürnkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the www, 1998. Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization.
- R. Ghani, S. Slattery, and Y. Yang. Hypertext categorization using hyperlink patterns and meta data. In *the Proceedings of the 18th International Conference on Machine Learning*, The Berkshires, Massachusetts, USA, 2001.
- M. Gibbon and E. Greevy. The truth about racism. SALIS Seminar Series, 2003.

- C. Goller, J. Löning, T. Will, and W. Wolff. Automatic document classification: A thorough evaluation of various methods. *IEEE Intelligent Systems*, 14(1):75–77, 2000.
- E. Greevy and A.F Smeaton. Classifying racist texts using a support vector machine (poster). In *the Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, UK, 2004a.
- E. Greevy and A.F. Smeaton. Text categorisation of racist texts using a support vector machine. In *the Proceedings of the 7th International Conference on the Statistical Analysis of Textual Data / Actes des 7es Journées Internationales d'Analyse Statistique des Données Textuelles*, pages 533–544, Louvain-la-Neuve, Belgium, 2004b.
- B. Grilheres, S. Brunessaux, and P. Leray. Combining classifiers for harmful document filtering. In *Proceedings of the RIAO'2004 on Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval*, Avignon, France, 2004.
- P.J. Hayes, P.M. Andersen, I.B. Nirenburg, and L.M. Schmandt. A shell for content-based text categorization. In *the Proceedings of the 6th IEEE Conference on Artificial Intelligence for Applications*, pages 320–326, Santa Barbara, California, USA, 1990.
- J. Hu, R. Kashi, and G. Wilfong. Comparison and classification of documents based on layout similarity, 2000.

- T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998a.
- T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *the Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, 1998b. Springer Verlag, Heidelberg, Germany.
- J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony. A framework for structural risk minimisation. *Computational Learning Theory*, pages 68–76, 1996.
- F. Kellely and A.F. Smeaton. Automatic phrase recognition and extraction from text. In *the Proceedings of the 19th Annual BCS-IRSG Colloquium on Information Retrieval Research*, Aberdeen, Scotland, 1997. Furner and D.J. Harper (Eds.), Springer Electronic Workshops in Computing.
- W. Kraaij and R. Pohlmann. Using linguistic knowledge in information retrieval, 1996.
- R. Krovetz. Viewing morphology as an inference process,. In *the Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–203, Pennsylvania, USA, 1993.
- H. Lechleiter and E. Greevy. The language of open racism: A corpus linguistic analysis. In *Societas Linguistica Europea Conference*, Lyon, France, 2003.

- D. D. Lewis. Evaluating text categorization. In *the Proceedings of Speech and Natural Language Workshop*, pages 312–318, Pacific Grove, California, USA, 1991. Morgan Kaufmann.
- D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In *the Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, Washington, USA, 1995.
- D.D. Lewis. Feature selection and feature extraction for text categorization. In *the Proceedings of Speech and Natural Language Workshop*, pages 212–217, San Mateo, California, USA, 1992. Morgan Kaufmann.
- D.D. Lewis and B.W. Croft. Term clustering of syntactic phrases. In *the Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–404, Brussels, Belgium, 1990.
- D.D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *the Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.
- C. Liao, S. Alpha, and P. Dixon. Feature preparation in text categorization. In *In Proceedings of the Australian Data Mining Workshop*, Canberra, Australia, 2003.
- P. Martin. Absolute relatives—the language of online racial identity. In *the*

- Proceedings of the 30th Annual Symposium of the Royal Irish Academy*, Dublin, Ireland, 2003a.
- P. Martin. So or also: Racist use of adverbial phrases. In *Societas Linguistica Europea Conference*, Lyon, France, 2003b.
- D. Mladenic and M. Globelnik. Word sequences as features in text learning. In *the Proceedings of the 17th Electrotechnical and Computer Science Conference*, Ljubljana, Slovenia, 1998.
- K.J. Mock and V.R. Vemuri. Information filtering via hill climbing, wordnet and index patterns. *Information Processing and Management*, pages 633–644, 1997.
- K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *the Proceedings of the 16th International Conference on Machine Learning*, pages 268–277. Morgan Kaufmann, San Francisco, CA, 1999.
- K.R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.
- T. Musuyama and H. Nakagawa. Two step pos for svm based text categorization. *Information Processing Technology for Web Utilization, IEICE Transactions on Information and Systems*, pages 15–22, 2004.
- PRINCIP Partners. Methodology for constituting suitable corpora, 2002. URL <http://www.princip.net>.

- J. Pierre. Practical issues for automated categorization of web pages, 2000.
- G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
- H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124, 1998.
- H. Schütze, D. Hull, and J.O. Pedersen. Comparison of classifiers and document representations for the routing problem. In *the Proceedings of the 15th Annual International ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, Washington, USA, 1995.
- F. Sebastiani. A tutorial on automated text categorisation. In Analia Amandi and Ricardo Zunino, editors, *the Proceedings of the 1st Argentinian Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, Argentina, 1999.
- F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. ISSN 0360-0300.
- A.F. Smeaton. Information retrieval: Still butting heads with natural language processing? In *Summer School on Information Extraction*, Lecture Notes in Computer Science, pages 115–138, Berlin, Germany, 1997.
- S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

- H. Sorensen, A. O’Riordan, and C. O’Riordan. Profiling with the INFOrmer text filtering agent. *J.UCS: Journal of Universal Computer Science*, 3(8): 988–999, 1997.
- C.M. Tan, Y.F. Wang, and C.D. Lee. The use of bigrams to enhance text categorization. *Information Processing Management*, 38(4):529–546, 2002. ISSN 0306-4573.
- V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
- J. Wilbur and K. Sirotkin. The automatic identification of stop words. *Journal of Information Science*, 18:45–55, 1995.
- R. Wodak and M. Reisigl. *Discourse and Discrimination. Rhetorics of racism and anti-Semitism*. Routledge: London and New York, 2001.
- Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *the Proceedings of the 17th Annual International ACM SIGIR Conference Research and Development in Information Retrieval*, pages 13–22, Dublin, Ireland, 1994.
- Y. Yang. Noise reduction in a statistical approach to text categorization. In *the Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 256–263, Seattle, Washington, USA, 1995.
- Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999. ISSN 1386-4564.

- Y. Yang. A study on thresholding strategies for text categorization. In *the Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 137–145, New Orleans, Louisiana, USA, 2001.
- Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, pages 252–277, 1994.
- Y. Yang and X. Liu. A re-examination of text categorization methods. In *the Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, Berkley, California, USA, 1999.
- Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorisation. In *the Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, Nashville, Tennessee, USA, 1997.
- Y. Yang and J. Wilbur. Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society Information Science*, 1996.