2007-01-01

Investigating Text Message Classification Using Case-based Reasoning

Matt Healy
Technological University Dublin

Investigating Text Message Classification Using Case-based Reasoning

Matt Healy

A thesis submitted to the Dublin Institute of Technology

in fulfillment of the requirements for the degree of

Masters of Philosophy

School of Computing

March 2007

Declaration

I certify that this thesis which I now submit for examination for the award of Masters of Philosophy, 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.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other Institute or University.

The work reported on in this thesis conforms to the principles and requirements of the Institute's guidelines for ethics in research.

The Institute has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis by duly acknowledged.

Signature [illegible] Date 22/10/07

Acknowledgements

I would like to express my deepest thanks and gratitude to my supervisor Sarah Jane Delany, for her guidance and advice, and for her continued support and encouragement throughout the duration of my research.

I would also like to thank the staff in the School of Computing in Kevin Street, who made me feel welcome and for their support and friendship. Thanks especially goes to Kuda and Essam, who made our office stimulating and enjoyable place to work and offered provocative conversations, intellectual stimulation and welcomed distractions.

Finally I would like to thank my family and friends who have supported me throughout the duration of this research. Thank you to all these people, for without which this would not have been possible

Matt Healy
Dublin Institute of Technology
March 2007

Abstract

Text classification is the categorization of text into a predefined set of categories. Text classification is becoming increasingly important given the large volume of text stored electronically e.g. email, digital libraries and the World Wide Web (WWW). These documents represent a massive amount of information that can be accessed easily. To gain benefit from using this information requires organisation. One way of organising it automatically is to use text classification. A number of well known machine learning techniques have been used in text classification including Naïve Bayes, Support Vector Machines and Decision Trees, and less commonly used are k-Nearest Neighbour, Neural Networks and Genetic Algorithms.

One aspect of text classification is general message classification, the ability to correctly classify text messages containing text of different lengths. There are many applications that would benefit from this. An example of such applications are, personal email filtering, filtering email into different categories of business and personal email and spam email and email routing, e.g. routing email for a helpdesk, so that the email reaches the correct person.

This thesis presents an investigation of applying a Case Based Reasoning (CBR) approach to general text message classification. Case-based Reasoning was chosen as it was found to perform well for a particular type of message classification, spam filtering. CBR was found to have certain advantages over other machine learning techniques such as Naïve Bayes. It was able to handle the dynamic nature of spam better than other machine learning techniques and offered the ability for the training data to be easily updated continuously and to have new training data immediately available.

The objective of this research is to extend previous work conducted on spam filtering to general message classification, which includes classifying short and long text messages into multiple categories. Short text message classification presents a particular challenge as the concept being learnt is weak. We investigated two types of similarity metrics used with CBR, feature based and featureless similarity metrics. We then compared CBR using both feature based and featureless similarity metrics with two well known machine learning techniques, Naïve Bayes (NB) and Support Vector Machine (SVM). These two machine learning techniques serve as base line classifiers as they seem to be currently the classifier of choice in the text classification domain. The results of this research show that CBR using a featureless similarity metric achieves better performance than CBR using a feature base similarity metric. The results also show that when using CBR with a feature based similarity metric the classification task required different feature types and different feature representations, depending on the domain.

We also investigated whether a case-base editing technique developed for spam case-bases improves the performance over unedited case-bases on different text domains. We found that the case-base editing technique used for spam filtering performs well for email based case-bases but not for other text domains of either short or long text messages.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is a plethora of textual information stored electronically, including email, digital libraries and the World Wide Web (WWW). These sources are constantly changing and growing on a daily basis, with new web pages being added to the WWW, people blogging and posting messages on message boards, millions of people sending emails every day and new scientific papers being published. This enormous volume of information is an inevitable consequence of the information age. These examples, among many, represent a massive amount of easily accessible information. To glean anything useful from this information, it has to be organised. Organisation of this information facilitates such tasks as storing files or documents into hierarchical structures or topic identification to support topic-specific processing operations. One method of organising this information is text classification. Text classification is defined as the "assignment of natural language texts to one or more predefined categories based on their content" (Sebastiani 2001).

There are many ways in which text classification can help with organising information such as using it to route and dispatch documents based on their content or using text classification to categorise web pages into hierarchical structures to aid search engines.

Manually performing text classification is costly, time consuming and labour intensive. It also has many disadvantages such as being prone to human error. Also specialised areas require a specialist which can be costly and people can disagree on how to organise the information. One way of automating this process is to use machine learning techniques.

There have been many machine learning techniques applied to the problem of text classification, some of the more popular are Naïve Bayes (Lewis 1998) and Support Vector Machines (Dumais et al. 1998). Both of these have been shown to be good at text classification (Katirai 1999, Lewis and Ringuette 1994). Other machine learning techniques such as Decision Trees (Alpaydin 2004) and Artificial Neural Networks (Sebastiani 2001) have also been applied to text classification (Calvo and Ceccatto 2000, Fred J. Damerau and Indurkhya 2002).

All these techniques have one thing in common, they are all *eager* learners. They build a generalised model from training examples, which is used to process new and previously unseen requests. These learners have some disadvantages such as they are not easily updateable. Every time new training examples are added, they require a retraining process to incorporate the new examples into the classification model. They also can be classified as *global* learners. Global learners try and create a single model that encapsulates all the diverse training examples. These learners can find classification difficult if the concepts they are learning are very diverse.

An alternative to using eager, global learners, which can require a lengthy training process and cannot handle very diverse training data, is to use lazy, local learners. Lazy learners postpone the decision of how to generalise beyond the training data until a new query instance is observed and local learners in effect build a local model for every request they receive. An example of a lazy, local learner is a case-based reasoner.

Cased Based Reasoning (CBR) (Mitchell 1997) is an instance based machine learning technique and it classifies new cases by analysing similar cases while ignoring cases that are very different from the new case. CBR is founded on the premise that similar problems have similar solutions and the same types of problems will occur over time.

CBR has been shown to perform well in the text domain of spam filtering. CBR was able to handle the diverse concept and able to manage the concept drift (Delany et al. 2005, 2004). This suggests that CBR is well suited for text classification when the domain is diverse and evolving.

This thesis presents an investigation of applying a case-based approach to general text message classification, which includes classifying short and long text messages into multiple categories, where the concepts being learned can be weak or diverse. Our investigation used two different types of similarity measures; a feature based similarity metric and a featureless similarity metric. For the feature based similarity metric, we used the traditional stages used in text classification namely feature extraction and representation, dimensionality reduction and classification. For the featureless similarity metric we processed the documents and used a general mathematical theory to calculate the similarity between documents thereby bypassing the feature extraction and representation and dimensionality reduction processes.

We apply our case-based approach to a number of datasets. The datasets are divided into datasets which contain short text messages, long text messages and a mixture of both. We will show in this thesis that the type of features and their representations differ with different text domains when using a feature based similarity metric. We also show that a case-base editing technique that was developed for editing a case-base of legitimate and spam email is not appropriate for other text domains. We will also demonstrate that a featureless similarity metric produces better results than a feature based similarity metric.

In this thesis we compared both our feature based and featureless CBR approaches with other machine learning techniques and ensemble methods. We show that CBR using a featureless similarity metric performs better than some of the most commonly used ensemble techniques, such as One Vs All and Round Robin.

1.1 Contributions of this Thesis

This thesis is concerned with the application of case-based reasoning for the purpose of general message classification. The main contributions of this thesis are the following:

(i) The case-base editing technique used for spam filtering does not perform well for classification of short and long text messages.

(ii) Depending on the domain, the classification task will require different feature types and different feature representations, when using a feature based Case-based Reasoning classifier.

(iii) Case-based reasoning with a featureless similarity metric achieves better performance than case-based reasoning with a feature based similarity metric on datasets that contain long text.

(iv) Case-based reasoning does not perform well for short text message classification.

(v) When using feature base similarity metrics, letter features are useful in general email classification and not just spam and non-spam

1.2 Associated Publications

The publication that is related to this thesis is:

- Matt Healy, Sarah Jane Delany and Anton Zamolotskikh (2005) An Assessment of CBR for Short Text Message Classification In: N. Creaney (ed.) Proceedings of the Sixteenth Irish Conference on Articial Intelligence and Cognitive Science (AICS 2005), p257-266.

1.3 Summary and Structure of this Thesis

This section outlines the rest of this thesis. Chapter 2, describes text classification and its various processes. It then describes two popular machine learning techniques used in text classification, Naïve Bayes and Support Vector Machines. It also describes Case-based Reasoning in detail using the 4-R model. This chapter also describes various similarity metrics used in textual CBR and also various ensemble techniques that can be used in text classification.

Chapter 3 describes the different datasets used and the design of the CBR system used to classify text messages and . It describes the feature extraction and representation, dimensionality reduction and classification processes used.

Chapter 4 discusses the various evaluations of the CBR system. These evaluations include determining the best case representation for a case-base using a feature based similarity metric and determining if CBE, a case-base editing technique developed for the spam filtering domain, is appropriate for other text domains. This chapter also compares a case-base using a featureless similarity metric and one with a feature based similarity metric.

Chapter 5 compares CBR with different ensemble techniques that can be used in text classification. It also compares CBR with two other machine learning techniques: Naïve Bayes and Support Vector Machines.

This thesis finishes with Chapter 6, which discusses the conclusions of this thesis.

Chapter 2

Background

Machine learning (ML) techniques have been applied to the text classification problem for many years. There are many different ML techniques used, two of the more popular are Naïve Bayes and Support Vector Machines. This chapter starts with a description of the various stages of text classification (section 2.1). It continues with a description of a Naïve Bayes classifier and a Support Vector Machine classifier and how they have been applied to text classification (section 2.1.3).

It then describes in detail a machine learning technique called Case-Based Reasoning (section 2.2), which is the technique applied to the problem of text classification in this thesis. This section explains the different stages in CBR and the advantages of using CBR for text classification. It then moves on to a description of the various different similarity metrics that can be used in textual CBR (section 2.3). This includes feature based and featureless similarity metrics.

The penultimate section discusses the different ensemble techniques used within the ML field (section 2.4) for text classification. The chapter finishes with the conclusions gleaned from this background review.

2.1 Text Classification

Text classification (TC) also known as text categorisation or topic spotting can be defined as the task of assigning natural language texts with a predefined set of thematic categories (Sebastiani 2001, 2002) or the automated assignment of natural language text to predefined categories based on their content (Lewis 1992).

A more formal definition of text classification is the task of assigning a Boolean value to each pair $(d_j, c_i) \epsilon D \times C$, where D is a set of documents and $C = c_1, \cdots, c_{|C|}$ is a set of predefined categories. A value of T (True) assigned to (d_j, c_i) indicates a decision to assign classification c_i to document d_j, while a value of F (False) indicates a decision not to assign classification c_i to document d_j. This can be described by the unknown *target function*:

$$\Phi : D \times C \rightarrow (T, F)$$

The need for TC stems from the vast amounts of mostly unorganized information, which is an inevitable consequence of the Information Age. This information normally requires some processing. There are many tasks to which text classification can be applied such as

(i) Word sense disambiguation, is the activity of finding the meaning of a given ambiguous word in its context. This has been suggested by Scott and Matwin (1999), Sebastiani (2001) and Granitzer (2003). This type of application could be incorporated into a software thesaurus.

(ii) Document routing or dispatching (Sebastiani 2001, Iyer et al. 2000) is the activity of classifying an incoming stream of documents, and deciding where they should be routed, depending on their content, e.g. help desk queries.

(iii) Web page classification under hierarchical Internet directories (Sebastiani 2001, Katirai 1999, Attardi et al. 1999). This is concerned with making the searching and classification of documents and web pages easier. It is suggested when web pages are classified hierarchically that it might be easier to traverse the hierarchical structure, instead of issuing a query to a general purpose Web engine.

These tasks would normally be quite difficult and time consuming if performed in a manual fashion. TC consists of a number of different stages. These stages are feature extraction and representation (see section 2.1.1), dimensionality reduction (see section 2.1.2) and classification (see section 2.1.3).

2.1.1 Feature Extraction and Representation

Feature extraction is concerned with the clean up and tokenization or extraction of words, letters and/or phases and the extraction of statistical or structural information from a document or group of documents. The more popular processes performed in the clean up operation are;

(i) Converting words to lower case and removing punctuation marks (Cohen and Singer 1996).

(ii) Removal of stop words (Fox 1990, 1992). Stop words are frequent words that carry no discriminatory power, words such as *the*. These words are normally functional or connective words that are assumed to carry no information content.

(iii) Word stemming, this reduces a word or term to its root form, mapping different morphological variants to a common stem (Frakes 1992). This is normally done to group words into the same conceptual meaning, for example *talking*, *talker*, *talked* and *talk* would all be mapped to the common stem *talk*. A well known stemming algorithm is the Porter algorithm (Porter 1997).

The removal of stop words and word stemming also reduce the size of the feature space, which can be extremely beneficial as some machine learning techniques do not scale well with large feature sets. There are other clean up operations that can be performed but are not as widely used, such as the removal of any tags, for example HTML or XML tags, although this information could be useful in certain text classification domains e.g. keeping HTML tags in spam filtering (Delany et al. 2004, 2005). Another clean up operation is the replacement of accented characters with non-accented characters (Delany et al. 2004). Depending on the domain, all or some of the clean up operations are performed. There is some evidence to doubt the universality of their effectiveness as Tang (2001) compared the exclusion and inclusion of both stemming and stop-words and found that it does not significantly affect the performance of a Support Vector Machine.

The next operation in feature extraction is normally tokenization; this helps to identify the possible lexical features. Different linguistic components of a document can form different types of features. There are many ways of tokenizing a document. A common method is to tokenize the document using the white spaces between words as breaks. This will produce a set of words or features also called *Bag of Words* representation of a document (Mitchell 1997, Lewis 1992, Lodhi et al. 2002, Robinson 2003). In this each token normally corresponds to a single word found in the training corpus. Scott and Matwin (1999) observed that a great deal of the information from the original document is discarded when using bag of words, as paragraphs, sentence and word order is disrupted, and syntactic structures are broken.

Another way of tokenizing a document is using phrases. The use of phrases instead of words was suggested, because words alone do not always represent true atomic units of meaning (Lewis 1992).

Documents can also be tokenized into single characters. The tokenization process can also produce certain statistical information also known as structural information such as the proportion of uppercase characters or the proportion of punctuation characters.

After tokenisation the contents of a document have to be represented in a logical way which can be processed by classification algorithms. A common way to represent these documents is the vector space model. A document is considered as possessing a set of features, which are the tokenised lexical elements which are found in the document. A vector is constructed wherein each component corresponds to a feature and has a value which reflects the degree to which the feature is associated with the document, $d_j = (f_{j1}, f_{j2}, \dots, f_{jn})$.

The features within the feature vector can have a binary or numeric representation. With binary feature representation the feature either exists or the feature doesn't exist in the document, this is often called the existence rule and is given in equation 2.1

$$f_{ij} = \begin{cases} 1, & \text{if } f_i \text{ is present in } d_j \\ 0, & \text{if } f_i \text{ is not present in } d_j \end{cases} \tag{2.1}$$

For numeric feature representation, the feature is represented by the frequency of the occurrences of the feature in the document. This is called term frequency weighting and there are a number term frequency weighting schemes, but most of these are based on two empirical observations regarding text: the more times a token occurs in a document, the more relevant it is to the topic of the document and the more times the token occurs throughout all documents in the corpus, the more poorly it discriminates between documents (Aas and Eikvil 1999).

The most common term frequency weighting scheme is *tfxidf* (Salton and Buckley 1988, Lam and Lee 1999) weighting where TF means *term occurrence frequency* and IDF means *inverse document frequency*, and is given in equation 2.2.

$$f_{ij} = f_{ij}^{n} log\left(\frac{N}{n}\right) \tag{2.2}$$

where N is the number of documents in the document corpus, n is the number of documents in which the i^{th} feature or term appears and f_i^n is the normalised frequency of the feature in the document. This takes into account the frequency of the word throughout all documents in the collection. It assigns a weight to word f_i in document d_j in proportion to the number of occurrences of the word in the document and inversely proportional to the number of documents in the collection for which the word occurs at least once (Aas and Eikvil 1999, Lam and Lee 1999). This weighting method is noted to be the most preferred weighting system for text categorization (Sebastiani 2001).

Another frequency weighting scheme is, tfc weighting (Salton and Buckley 1987, Siroker and Miller 2003), this is similar to *tdxidf* weighting but takes into account the different lengths of documents and is given in equation 2.3.

$$f_{ij} = \frac{f_{ij}^{n} log\left(\frac{N}{n}\right)}{\sqrt{\sum_{j=1}^{M}\left[f_{ij}^{n} log\left(\frac{N}{n}\right)\right]^2}} \tag{2.3}$$

This uses length normalisation as part of the word weighting formula, thereby eliminating the effects of differences in document length. A weighting algorithm very similar to tfc is LTC weighting except that it takes the logarithm of the frequency of the feature, in an attempt to smooth the weights. In this way, words with extremely high frequency won't dominate. For example there shouldn't be a greater difference between a word appearing 50 times and 51 times than one appearing no times and one time.

An alternative weighting scheme is Entropy Weighting (Siroker and Miller 2003), which uses the average uncertainty or entropy of a word (feature) in question in the documents. This quantity is 1 if the word is equally distributed over all documents and 0 if the word occurs in only one document. However this weighting scheme was found to be computational complex for processing a large corpus of documents (Siroker and Miller 2003).

2.1.2 Dimensionality Reduction

The high dimensionality of the feature space, resulting from feature extraction and feature representation is problematic for the text classification task. Many classification techniques cannot deal with such large feature sets. Processing these large feature sets is extremely costly in computational terms.

Dimensionality reduction is the process of reducing the size of the feature space from a large set (tens of thousands of features) to a much smaller number. This also reduces the tendency of the classier to overfit, (i.e. the tendency of the classifier to better classify the data it has been trained on than new unseen data), and to make the task more manageable for the classifier, as many learning methods do not scale well with large problem sizes (Sebastiani 2001). Dimensionality reduction can take one of three forms, Feature Selection, Feature Transformation and Wrappers.

Feature Selection

Feature selection is the process of selecting from the original set of features a subset of features that are most useful or suitable for compactly representing the meaning of the documents, according to some criteria (Galavotti et al. 2000). There are many feature selection techniques.

A common technique is Information Gain (IG) (Mitchell 1997). This is an information-theoretic function which measures the amount of information obtained for category prediction by knowing the presence or absence of a term in a document. It was found to be effective in aggressively removing features without losing classification accuracy (Yang and Pedersen 1997). IG measures the expected reduction of entropy caused by partitioning the instances according to feature f as shown in equation 2.4, where $values(f)$ is the set of all possible values for feature f. S is a set of training instances and S_v is a subset of S in which f has value v.

$$IG(S,f) = Entropy(S) - \sum_{v \epsilon values(f)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2.4}$$

Entropy is defined as a measure of how much randomness or impurity there is in a dataset and is given in equation 2.5, where p_i is the proportion of S belonging to class i and c is the number of classes.

$$Entropy(S) = \sum_{i=1}^{c} -p_i log p_i \tag{2.5}$$

Another algorithm used for feature selection is Mutual Information (Yang and Pedersen 1997), which is an iterative process of identifying significant lexical patterns by examining the statistical frequencies of word co-occurrences. Mutual information is a metric that measures how frequently a pattern happens in the corpus, relative to its sub-patterns and is given in equation 2.6.

$$MI(C,f) = \log \frac{P(c_i|f_t)}{P(c_i)P(f_t)} \tag{2.6}$$

$P(c_i)$ is the probability of the document belonging to class c_i, $P(f_t)$ is the probability of feature f_t occurring in a document and $P(c_i|f_t)$ is the probability of feature f_t occurring in class c_i.

Odds Ratio (OR) ranks documents according to their relevance for the positive class using occurrences of different words as features (Caropreso et al. 2001). OR was found to give a high accuracy with a small subset of features (Mladenic 1998). OR is given in equation 2.7.

$$OR(f, c_i) = \frac{P(c_i|f_t)\Big[1 - P(\bar{c}_i|f_t)\Big]}{\Big[1 - P(c_i|f_t)\Big]P(\bar{c}_i|f_t)} \tag{2.7}$$

where $P(c_i|f_t)$ is the probability of feature f_t occurring in class c_i and $P(\bar{c}_i|f_t)$ is the probability of feature f_t not occurring in class c_i.

One of the simplest feature selection techniques is Document Frequency Thresholding, where the document frequency of each feature in the training set is computed and the features whose document frequency is less than some predefined threshold are removed (Yang and Pedersen 1997). This feature selection techniques has one of the lowest costs in computation but assumes that rare features either hold no useful information for category prediction or are not influential in global performance.

Feature Transformation

Feature transformation is a process that constructs a set of new features from an original term set through some functional mapping, while retaining as much information as possible (Huang et al. 2002).

A commonly used technique in feature transformation is Latent Semantic Indexing (LSI), which compresses document vectors into document vectors of a lower-dimensional space whose dimensions are obtained as combinations of the original dimensions by looking at their patterns of co occurrence (Huang et al. 2002, Aas and Eikvil 1999, Wang et al. 2005). LSI creates a matrix representation of training instances with rows corresponding to features and columns to documents. Then LSI uses a method of matrix decomposition called Singular Value Decomposition (SVD) to create a compressed matrix (Keller and Bengio 2003, Cunningham and Dahyot 2004). This is given in equation 2.8

$$M = USV \tag{2.8}$$

where M is the original matrix, U is the original matrix M decomposed into a reduced rank term matrix. S is a diagonal matrix of singular values and V is a document matrix. The row vector of matrix U and the column vector of matrix V are the projections of feature vectors and document vectors into singular value space. This produces a feature space that is more compact compared to the original (Huang 2003).

Another feature transformation technique is principal component analysis (PCA) also called the Karhunen-Lo'eve transform and is a commonly used linear projection method (Lam and Lee 1999). PCA is a statistical technique for dimensionality reduction that tries to minimize the loss in variance in the original data than that of the constructed set of new features.

Wrappers

In the wrapper approach the feature selection algorithm exists as a wrapper around the classification algorithm. The feature selection algorithm conducts a search though the feature space using the classification algorithm to evaluated candidate subsets. It then estimates the accuracy of the classifier based on that subset (Kohavi and John 1997). This is more computationally complex than feature selection and transformation techniques and becomes impractical for large features sets and therefore not suitable for text classification. Wrapper techniques also have the tendency of over fitting the training data (Cunningham and Dahyot 2004).

2.1.3 Classification

The last stage in text classification is the actual classification process which is performed by the classifier. There are two ways of using a text classifier which are Category Pivoted Classification and Document Pivoted Classification (Sebastiani 2001). Category Pivoted Classification is where given a category a system can find and retrieve all documents that should be filed under it, this is more on the information retrieval discipline side of text categorization. Document Pivoted Classification is where given a document, a system can find the category or categories which it should be filed under, this method is normally associated with the Machine Learning discipline (Sebastiani 2001).

The classifier's decision can either be a hard decision or a ranked decision. A hard decision is when a document is either inside or outside of a category, while a ranked decision is where a document is ranked by its appropriateness to certain categories.

There are many machine learning techniques used for text classification including Naïve Bayes, Support Vector Machines or k-Nearest Neighbour.

Naïve Bayes

Naïve Bayes (NB) is a probabilistic classifier based on Bayes Theorem (Mitchell 1997), which can manage high dimensionality. NB is naïve in a sense that it assumes that the effect of a feature value on a given class or category is independent of the values of other features. This assumption is called class conditional independence. In the case of text categorization, some words are statistically more likely to best define a category while others are not. The probability of a document belonging to a category can be calculated by combining the probabilities of certain words being able to best define a category. NBs implementation is given in equation 2.9, where each document is labelled as one of a set of classifications $c_i \in C$ and is described by a set of attributes $\{a_1, a_2, \ldots a_n\}$, where a_i indicates the presence of that attribute in the document.

$$c_{NB} = argmax_{c_i \in C} P(c_i) \prod_j P(a_j|c_i) \tag{2.9}$$

This implementation incorporates a Laplace correction in the conditional probabilities to prevent zero probabilities dominating (Niblett 1987). This is given in equation 2.10, where n_{ki} is the number of values for attribute a_i and $f = \frac{1}{m}$ where m is the number of training documents.

$$P(a_i|c_j) = \frac{n_{ij} + f}{n_j + f n_{ki}} \tag{2.10}$$

NB has been used in text classification in the area of spam filtering. Katirai (1999) compared NB and genetic algorithms for email spam filtering and they found that NB out performed generic algorithms but only marginally as they state "in fact the two performance measures are so close that we cannot be sure that the differences are statistically significant." In other text classification research NB has been compared with many other machine learning techniques. Lewis and Ringuette (1994) compared NB with decision tree algorithm and was found it was comparable to the decision tree algorithm. They also showed that the feature selection process is extremely important with regards to a classifier's performance.

NB has also been compared to an ensemble technique called Error Correcting Output Coding (ECOC) (Rennie 2001a) on multi-category text problems. ECOC has been traditionally used in the correction of errors in the transmission of data over a communication channel. In machine learning, ECOC is an ensemble technique where each member of the ensemble computes a single bit of a codeword which corresponds to class. Rennie showed that NB performs well when there were enough examples in the training set to learn from. Rennie also showed that NB does not perform as well as ECOC, as long as ECOC had enough examples in the training set for each member of the ensemble. In situations where the number of examples available to the classifier is limited, ensemble methods such as ECOC are infeasible as there are too few examples for each ensemble member to learn from. In such cases a single learner might have enough examples to effectively model the concept being learnt.

Li and Jain (1998) compared NB with a k-nearest neighbour classifier, a decision tree classifier and a subspace method. The subspace method decomposes the feature space into m sub-regions of lower dimensionality, where each region is a representative feature space for a corresponding class. They found that NB outperforms the other classifiers and noted that the NB classifier performance improved as the number of features used increased.

Support Vector Machine

A Support Vector Machine (SVM) (Mitchell 1997, Joachims 1999) is a linear classifier, which in its simplest form is a hyperplane or decision surface that separates positive and negative examples with maximum margin. The margin is defined by the distance of the decision surface to the nearest of the positive and negative instances (Dumais et al. 1998). The instances closest to the decision surface are called the "support vectors" as they support the decision surface on both sides of the margin. SVM uses a kernel function that transforms the dimensionality of the feature space into a higher dimensionality, which is linearly separable.

There are two types of SVM a hard margin SVM and a soft margin SVM. Hard-margin SVMs try to solve the classification problem by constructing a decision surface that completely separates the positive and negative instances. This can be difficult or impossible if there is noise in the data. To solve this problem, a soft-margin SVM allows for some instances to fall within the margin or on the wrong side of the decision surface.

Tang (2001) evaluated using SVM for text classification and found that the larger the training set the better the classification accuracy. Tang also evaluated feature extraction and selection techniques using SVMs and found that the use of stop-word removal and inverse document frequency weighting lead to a slight increase in classification accuracy.

Fukumoto and Suzuki (2002) took advantage of an SVMs ability to handle large feature spaces to help classify instances in large text corpora. They used SVMs with NB, letting the NB classifier select the training data for the SVM, which helped to reduce the computational complexity associated with the SVM. They compared this hierarchical classifier with an ensemble technique and found that there was no significant difference in performance.

Dewdney et al. (2001) compared an SVM with NB and C4.5 (Quinlan 1993), a decision tree algorithm, on the classification of documents into different genres. They found that the SVM performed better than the other classifiers. They suggested this was due to the SVM's ability to allow for an *unknown classification* and the NB assumption that features are independent.

Other Machine Learning Techniques used for Text Classification

This section describes some of the less common machine learning approaches to text classification. These techniques are normally experimental. The techniques described here are Artificial Neural Networks and Decision Trees.

Artificial Neural Networks (ANN) (Mitchell 1997) are a computational model that consists of a number of simple processing units that communicate by sending signals to each other over a large number of weighted connections arranged into a network, which have the ability to learn complex nonlinear input-output relationships (Wiener et al. 1995). The inputs units represent terms, the output units represent the category or categories of interest and the weights between the units represent dependency relationships. Each unit or neuron has an activation function and a threshold value. The sum of the inputs and weights are calculated and are used as the input to the activation function. If the result of the activation function is greater than the threshold value, the unit is activated, i.e. an output is generated. To classify a text document, its terms are feed into the input units, the activation of these units is calculated and the results are propagated forward through the network, and the value of the output units determines the category.

Calvo and Ceccatto (2000)'s research compared ANN with SVM, NB and k-nearest neighbour for the classification of the Reuter newswire text corpus (Lewis 1999). They found that ANN performed as well as SVM and k-nearest neighbour but noted that the ANN's performance has a degree of randomness, inherent to the learning process and due to the ANN's initial conditions. Due to this randomness, ANN can have a problem of user acceptance, where users are sceptical of using and trusting it when they have no idea of how the ANN was performing the classification.

A Decision Tree (DT) (Mitchell 1997) is a hierarchical data structure that implements a divide and conquer strategy, which can be used for classification (Alpaydin 2004). A decision tree is composed of internal decision nodes and terminal leaves. Each decision node implements a test function with discrete outcomes labelling the branches. A decision node is given an input, performs a test function on it and depending on the outcome of that function one of the branches is taken or outputted. In a text classifier DT the internal nodes are labelled by terms, branches departing from them are labelled by tests on the weight determined by the number of occurances of the term has in the test document and the terminal leaves are labeled by categories (Sebastiani 2001). In this classifier a document is categorized by recursively testing the weights of the terms labelling the internal decision nodes and comparing them to the vector space from the test document until a leaf node is reached. The category label of the leaf node is assigned to the test document.

Fred J. Damerau and Indurkhya (2002) used decision trees instead of a linear classifier as the tree can be reduced to a set of understandable rules for classification, which allow for manual adjustment if necessary. They used DT for the classification of a large corpus of text documents and found its performance is comparable to other well known machine learning paradigms, such as NB.

The next machine learning technique described is Case-based Reasoning. In this thesis we evaluated this technique for the classification of text messages and compare it with the other popular techniques. The next section will discuss CBR in detail, it will describe the various stages, previous research in CBR and why we think CBR is well suited to text classification.

2.2 Case-based Reasoning

Case Based Reasoning (CBR) (Mitchell 1997) is an instance based machine learning technique. It solves new problems by re-using or adapting solutions to previous problems that it has already come across and that are similar to the new problem. This technique stores previous problems as *cases*, which contain a description of the problem and their solutions. These cases are kept in a knowledge store called the $case - base$, which is called upon to solve new problems.

CBR is based on two tenets about the natural world, the first is that similar problems have similar solutions and therefore solutions for previous problems are useful as possible solutions or part solution to new problems. The second tenet is that an intelligent agent or system will encounter the same types of problems over time, therefore future problems may be similar to current problems (Leake 1996).

The CBR method has developed from two different disciplines. The first is cognitive science, in a desire to model human behaviour as studies of human reasoning has demonstrated that humans reason from previous cases (Leake 1996). The second is machine learning and a desire to create AI systems that do not use generalised rules and can be tailored to be more effective (Aamodt and Plaza 1994).

There are many strengths of CBR, one is that it can work well in domains that are not well understood and another is it can solve complex problems with limited knowledge or experience. It also has other advantages over other machine techniques; CBR is a lazy and local learner. CBR as a lazy learner, defers the decision of how to generalize beyond the training data until a new case or query instance is observed. This means CBR does not have a training phase or stage before being able to solve new problems. The local learner aspect of CBR allows it to select only relevant examples to solve a new problem. This means it effectively builds a local model for each request that it processes. This allows CBR to handle domains where there is a lack of homogeneity among the training examples. A global learner would try and create a model that encapsulates all the diverse training examples, where a local learner would just use the training examples that were most relevant to a new problem. The knowledge store, the case-base is also easily updatable. New cases can be added to the case-base without the need for any re-training. Old cases also be removed easily from the case-base.

CBR can be described as four sub processes (Aamodt and Plaza 1994).These are

(i) **Retrieve**: the most similar case or cases in the case-base are found and retrieved.

(ii) **Reuse**: the information and knowledge in the retrieved cases is used to propose a solution to the new problem.

(iii) **Revise**: the proposed solution is evaluated.

(iv) **Retain**: the useful parts of the solution are stored to be used in future problem solving.

Each of these will be explained in more detail see section 2.2.2-2.2.5, but before these are explained, the case representation has to be discussed as CBR is heavily dependant on the structure of the case-base.

2.2.1 Case Representation

As CBR solves problems by recalling previous experiences, the searching and matching processes have to efficient and effective. As new cases can be added to the case-base, there is a problem of how to represent a case, what will it store and how will it be indexed and organized (Aamodt and Plaza 1994). A conceptual view of a case contains;

(i) A description of the problem or situation.

(ii) A description of the solution.

(iii) A description of the outcome after the solution is applied.

The concepts of problem and solution have no general format and vary from application to application. When dealing with a classification problem, the solution is the class to which a case belongs.

In general a case is represented as a set of features, where the features are the characteristics of the specific domain and the problem that are judged to be most significant in determining the solution. The identification of the features is normally performed in a knowledge acquisition process. This process involves common data gathering techniques such as interviewing experts in the domain in question.

When applying CBR to text classification, the description of the problem is the features of the document (i.e. words and/or letters), the description of the solution is the classification of the document and the description of the outcome does not apply to classification tasks.

An implementation of a generic case representation which is becoming increasingly popular is in eXtensible Mark-up Language (XML). Hayes et al. (1998) proposed a XML based language called CBML (Case-Based Markup Language), which contains two CBR objects the case structure and the case content. The case structure defines the hierarchy and cardinality of the features of a case. The case content contains the case-base information in XML format.

2.2.2 Case Retrieval

The first process in CBR is the retrieval of similar cases. This process starts with a problem description (a new case) and ends with a best matching existing case or cases. The two main retrieval techniques used in CBR are the decision tree algorithm and the k-NN algorithm.

The decision tree algorithm (Wess et al. 1994) organises the features into a tree structure with the most discriminating features at the top of the tree. All cases in the case-base are organised using this tree structure. To retrieve the most similar cases the retrieval algorithm searches the structure matching nodes against the new case. There are some disadvantages of this method of case retrieval. The first disadvantage is the time taken to retrieve, which increases logarithmically with the number of cases. The second disadvantage is that decision trees have poor performance when there are missing features in the case representation. In addition it requires a significant number of cases to be able to identify the appropriate hierarchical structure.

The second algorithm is the k-NN algorithm. The principle on which the k-NN operates is that instances are classified based on the class of their nearest neighbours. The k-NN algorithm compares each case in the case-base with the target case and computes a similarity measure for each case. Typically the similarity measure is based on how similar the features of the target case are to cases from the case-base. Each feature is compared and a score is calculated depending on the similarity of the two features, the more similar the cases the higher the score. Features can be assigned weights depending how relevant they are to the solution of the case. k-NN returns the k number of cases with the highest similarity score. The solutions of the k cases are then used to derive a solution for the target case.

To improve the search time and reduce the computational complexity of the k-NN algorithm, an alternative similarity retrieval algorithm based on Case Retrieval Nets (CRNs) was conceived (Lenz et al. 1998). This algorithm produces the same results as k-NN. A CRN is a memory structure that borrows ideas from neural networks and associative memory models which allows an efficient and flexible retrieval of cases. CRNs are made up of three components. The first component is the Information Entity Nodes (IEs), which represent feature-value pairs within cases. The second component is the Relevance Arc, which links case nodes with the IEs that represent them. These can have weights that capture the importance of the IE. The final component is the Similarity Arc which connects IEs that refer to the same features. These can have weights relative to the similarity between connected IEs.

To classify a case, the target case is connected to the net by the use of relevance arcs. The activation is then spread across the net. Cases in the case-base accumulate an activation value relevant to their similarity to the target case. The higher the activation value, the more similar a case is to the target case.

2.2.3 Case Reuse

The next process of CBR is the reuse of case(s). This process uses the solution of the retrieved case or cases and adapts the solution(s), if necessary, to solve the target case's given problem. There are two different ways of reusing a previous case to solve a new problem; copy and adapt.

The simplest way to use a retrieved case in situations where the retrieved case is identical to the target case is to copy the unchanged solution of the retrieved case as the solution to the new problem (Leake 1996, von Wangenheim 2000). However the retrieved case is normally only similar to the given problem, and will rarely be directly applicable due to the differences between the query and the problem description of the case. As a result the retrieved case will normally be adapted in some way.

Adaptation techniques can be categorised as transformational adaptation and derivational adaptation (Aamodt and Plaza 1994, Roth-Berghofer 2003). Transformational adaptation uses the past case solution indirectly by applying transformation operators to the old solution. These operators are normally rules and formulae which adjust parameters accordingly. This normally requires additional domain knowledge. This can be obtained though user intervention. Derivational adaptation analyses how the problem was solved in the retrieved case. The method that was used to solve the retrieved case is then reused to solve the new problem case.

The complexity of the adaptation technique can cause a number of problems. The higher the complexity, the more computationally expensive it becomes. It has also been argued that complex adaptation is knowledge intensive and the knowledge required for defining transformational rules is not readily available as CBR is used in domains that are not well defined or understood (Watson 1998).

2.2.4 Case Revision

The third process of CBR is case revision. This process comprises of two steps, the evaluation of the solution and its diagnosis and if necessary, the repair of the solution using domain-specific knowledge.

The evaluation process applies the suggested solution and judges how well the solution solves the problem. The evaluation can be based on feedback from the real world. This can be done by applying it to a simulation program or internal model or asking an expert. The result of the evaluation could recommend that further adaptation or repair is necessary.

The repair process uses general domain knowledge about how to compensate for the causes of errors in the domain. The revised solution can then be retained if the revision stage has assured its correctness or it can be evaluated and repaired again until a satisfactory solution is achieved.

2.2.5 Case Retention

The final process of CBR is case retention. This process incorporates the useful parts from the new problem solving episode into existing knowledge. This process involves selecting information from the new case to retain, the form in which it should be retained, how to index the case for later retrieval, and how to integrate the new case in the case-base. This process is essentially the case-base's ability to learn by past experience. Successful case solutions are added to the case-base to smooth the progress of similar problems in the future. Repaired case solutions can be added to the case-base to help avoid repeating the same mistakes, i.e. the case-base learns from its mistakes.

2.2.6 Case Base Editing

Over the course of time a case-base can become increasingly large due to new cases being added. Case-base editing can help to maintain a case-base of reasonable size while preserving or even improving performance. The main objective of case-base editing is to reduce the size of the case-base by removing cases that do not contribute to or that impede the classification process. Case-base editing techniques can be *incremental*, where selected cases are added from the training set to a new initially empty case-base and *decremental*, where selected cases are removed from the training set.

Case-base editing techniques can be categorised into two types: competence preservation and competence enhancement (Brighton and Mellish 2002). Competence preservation (or redundancy reduction) is the process of removing redundant cases from the case-base. Redundant cases are cases that are in a cluster of cases that all have the same classification and therefore can be removed without affecting the case-base's generalisation accuracy.

Competence enhancement techniques are techniques that remove noisy or exceptional cases from the case-base. Noisy cases are cases that are mislabelled, while exceptional cases are cases that are in a cluster of cases of a different classification. An example of a competence enhancement technique is the Edited Nearest Neighbour (ENN) algorithm by Wilson (1972), which tries to identify noise by identifying cases that are not classified correctly with their k nearest neighbours.

Other types of case-base editing techniques can be considered a hybrid of both competence preservation and competence enhancement techniques such as Competence Based Editing (CBE) (Delany and Cunningham 2004). CBE is an editing technique that was developed for text based case-bases, though it could easily be used in other domains. This editing technique was found to lower the generalisation error on a case-base comprised of legitimate and spam email. CBE is comprised of two stages, the first is called Blame Based Noise Reduction (BBNR) and the second is called Conservative Redundancy Removal (CRR).

BBNR builds on the competence modelling ideas of Smyth *et al* (Smyth and Keane 1995, Smyth and McKenna 1998). BBNR not only indicates how well a case contributes to correctly classifying other cases in the case-base but also includes the notion of blame or liability. BBNR measures how often a case is the cause of or contributes to other cases in the case-base being misclassified and uses this information to remove the offending cases.

The second stage, CRR, identifies cases that are redundant and are to be removed. This stage tries to conservatively reduce the case-base, as aggressive techniques remove too many cases resulting in a loss in generalisation error as was found in the spam filtering domain (Delany and Cunningham 2004). It was found that editing a case-base using CBE yields the best generalisation accuracy in the spam filtering domain (Delany and Cunningham 2004).

2.2.7 Advantages of CBR for Text Classification

There are a number of advantages case-based reasoning offers over other machine learning techniques for text classification. The generic advantages which CBR offers are that it is a lazy learner, it defers the decision of how to generalize beyond the training data until a new case or query instance is observed, in contrast to eager learners which construct a model from the training instance before any target instance is seen. This is beneficial as there is no need for a time consuming training stage, which can also be computationally expensive because of the size of the feature space inherent in text domains. The down side to this is that all training instances have to be available for every request, but this limitation is diminishing with the rapid evolution of computer hardware (Delany 2006).

CBR also allows new cases to be added with ease, and does not require a learning stage to incorporate the new document into the model like other machine learning techniques. This is favourable as text domains are constantly growing and evolving (e.g. email). Another side to this is that CBR allows the case-base to be edited, thereby removing redundant and unhelpful cases that do not aid the classification process.

CBR is also a local learner. It selects appropriate and relevant training documents and effectively builds a local model for each request that it processes. This offers further advantages over eager techniques in domains where there is a lack of homogeneity among the training examples (Atkeson et al. 1997). This gives CBR an advantage in domains where the concept being learnt is diverse, such as the classification of message board postings on the Internet. Another advantage of local learners is that they do not suffer from data interference (Atkeson et al. 1997), where new documents being added to the training set in one area do not affect performance in others.

All of these advantages are useful when using CBR for text classification. Text domains are constantly changing, from the concept drift inherent in spam filtering, or the new use of language heralded by the advent of text messaging on mobile phones or instant messaging application (e.g. MSN Messenger). Text domains are normally diverse, from different emails sent to a technical helpdesk to newsgroup and message board postings on the Internet.

Textual CBR is the name given to CBR where the cases represent text documents. Some examples of these text domains are customer support and help desk. Lenz et al. (1998) used textual CBR for a help desk application which retrieves past problems with their solutions. By extracting problem identifiers similar cases can be used to solve the new problem. CBR offers some advantages such as the ability of incorporating domain specific knowledge into the case representation to aid the search of similar documents. This can be more efficient than the information retrieval approach of singling out relevant documents.

Another example of textual CBR is Ashley and Aleven (1991)'s system which teaches law students to argue using legal cases. The system was able to retrieval relevent cases from a legal case library for an argument and counter argument. They hoped to provide law students with a conceptual model for the criteria of selecting and describing precedents.

The extraction and creation of cases from text documents is vital in searching and retrieving similar cases. The representation of cases in textual CBR can be more complicated than the more traditional information retrieval methods used (e.g. bag of words), by incorporating domain specific knowledge or using more complex features (Bruninghaus and Ashley 2001).

2.3 Similarity Metrics used in Textual CBR

Similarity measures are concerned with determining the degree of similarity between two instances. In text classification the instances are document vectors, which consist of a set of features identified from a given document (see section 2.1.1). Similarity measures can be divided into two different types: feature based similarity metrics, where the similarity is calculated by using the features of each instance and featureless similarity metrics, where the similarity is calculated by a mathematical theory of similarity that uses no background knowledge or feature specific information.

2.3.1 Feature Based Similarity Metrics

There are many feature based similarity metrics used in text classification, ranging from those simple to implement to more complex measures. Normally in text classification, the training dataset D is made up of $(x_i)_{i\in[1,|D|]}$ instances. The training instances are described by a set of features F and each instance is labelled with a class label $y_j \in Y$. The intention is to classify an unknown instance q. For each $x_i \in D$:

$$Sim(q, x_i) = \sum_{f\in F} w_f \delta(q_f, x_{if}) \tag{2.11}$$

Where w_f is the feature relevance weight and $\delta(q_f, x_{if})$ is the feature specific similarity measure. There are a number of methods of calculating the similarity measure $\delta(q_f, x_{if})$. Some of these methods are described below.

Coyle et al. (2002) categorized similarity measures into discrete and difference based measures. The discrete similarity measure is where the value given for the similarity is either 1 or 0 depending on the values of the features being compared. A discrete similarity measure is given in equation 2.12, where similarity is 1 if the feature values are equal and similarity is 0 if the feature values are not equal

$$\delta_f(q_f, x_{if}) = \begin{cases} 1, & q_f = x_{if} \\ 0, & q_f \neq x_{if} \end{cases} \tag{2.12}$$

The Difference based similarity measure is calculated by computing the mathematical difference between two features. This metric is only suitable for numerically represented features. The difference based similarity measure is given in equation 2.13.

$$\delta_f(q_f, x_{if}) = |q_f - x_{if}| \tag{2.13}$$

Another way of measuring the likeness between cases is to use a distance measure. Distance measures are the inverse of similarity measures. They measure the distance between cases. A common distance measure is the distance between two points in Euclidean space which is implemented at the case level. The Eucilidean distance is given in equation 2.14

$$Sim(q, x_i) = \sqrt{\sum_{f \in F} (q_f - x_{if})^2} \tag{2.14}$$

A popular similarity measure in text classification and information retrieval is the cosine similarity measure (Salton and McGill 1986). This measure is based on angular distance and calculates the cosine of the angle between two instance vectors. The cosine similarity measure is given in equation 2.15.

$$Sim(q, x_i) = \frac{\sum_{f \in F}(q_f \cdot x_{if})}{\sqrt{\sum_{f \in F} q_f^2}\sqrt{\sum_{f \in F} x_{if}^2}} \tag{2.15}$$

2.3.2 Featureless Similarity Metrics

A featureless similarity measure calculates a similarity between two instances based on a general mathematical theory that uses no feature specific information. The two featureless similarity measures explained here are based on data compression. Data compression has been used as the basis of a similarity measure in domains other than text classification such as bioinformatics (Loewenstern et al. 1995) and clustering of time-series data (Keogh et al. 2004).

The two featureless similarity metrics describe here, formulated by Keogh et al. (2004) and Li et al. (2003) and are based on and inspired by the theory of Kolmogorov complexity (Kolmogorov 1965, Li and Vitanyi 1993). Kolmogorov complexity is a measure of the randomness of strings based on their information content.

The Kolmogorov complexity $K(x)$ of a string x can be is defined as the length of the shortest program capable of producing x on a Turing machine. The conditional Kolmogorov complexity $K(x|y)$ of x relative to y can be defined as the size of the smallest Turing machine capable of outputting x when given y as an input (Kolmogorov 1965). This can be used as a basis for a similarity measure, for example if $K(x|y) < K(x|z)$, then y is more similar to x than z is, because y contains more information content that is useful to outputting x than z. Kolmogorov complexity is not computable but it can be approximated by letting $K(x)$ be the best achievable compression for x

The first featureless metric described is a distance measure proposed by Keogh et al. (2004) which is called Compression-based Dissimilarity Measure (CDM). CDM is given in equation 2.16.

$$CDM(d_i, d_j) = \frac{C(d_i, d_j)}{C(d_i) + C(d_j)} \tag{2.16}$$

where $C(d_i, d_j)$ denotes the compressed size of the concatenation of d_i and d_j , $C(d_i)$ denotes the compressed size of d_i , and $C(d_j)$ denotes the compressed size of d_j. The maximum value CDM could produce in principle is 1, when d_i and d_j are so different that $C(d_i, d_j) = C(d_i) + C(d_j)$ and the lowest value this measure could produce is slightly above 0.5, even if $d_i = d_j$. Keogh et al. (2004) showed that data mining algorithms using a featureless similarity measure could compete with and outperform feature based algorithms. They also noted in the data mining domain, feature based algorithms are prone to overfitting.

The second featureless metric was proposed by Li et al. (2003) and is called the Normalized Compression Distance (NCD). NCD is given in equation 2.17

$$NCD(i, j) = \frac{C(d_i, d_j) - Min\{C(i), C(j)\}}{Max\{C(i), C(j)\}} \tag{2.17}$$

where $C(d_i, d_j)$ denotes the compressed size of the concatenation of d_i and d_j , $C(d_i)$ denotes the compressed size of d_i , and $C(d_j)$ denotes the compressed size of d_j. Li et al. (2003) showed that this similarity metric can be used to successfully classify natural languages and that it is not restricted to a particular domain. NCD has some advantages over CDM, in that it satisfies the identity axiom, the triangle inequality and the symmetry axiom, though this does not hinder CDM being applied to the classification task as long as the classifier does not rely on any of these properties.

This section has shown that there are many similarity measures used in textual CBR. Some based on the features in the documents while others are based on general mathematical theory.

Section 2.4 describes some ensemble techniques used in text classification. These techniques could all use the similarity metrics that were described in this section if the ensemble technique is using CB based classifiers.

2.4 Ensemble Techniques

Although real-world classification problems often have multiple classes, many learning algorithms are only able to differentiate between two classes i.e. they are inherently binary, for example SVMs. The solution to this problem has two principal approaches: firstly is to generalise the learning algorithm for multi-category problems or secondly to reduce the multi-category problem into a series or ensemble of binary problems, thereby enabling the binary learning techniques to solve the problem.

Classifier ensembles or classifier committees are based on the idea that the whole is greater than the sum of its parts, in this case the *parts* are the individual base classifiers and the *whole* is the collection of all the classifiers in the ensemble. In other words; the n different classifiers $\{c_1, c_2, \ldots c_n\}$ may be better than one if their individual judgments are appropriately combined (Sebastiani 2002).

This section is divided into two subsections. The first section is centered on the ensemble generation technique, where the second section describes the different ways of combining the outputs of the base classifiers in the ensemble.

2.4.1 Ensemble Methods

This section describes a variety of different ensemble techniques which are;

(i) Bagging Ensemble which trains each base classifier on a random redistribution of the training set.

(ii) Boosting Ensemble creates a series of classifiers, where each classifier in the series depends on the performance of the preceding classifier.

(iii) Round Robin Ensemble creates one classifier for each pair of classes.

(iv) One Vs. All Ensemble which constructs N binary classifiers from an N class problem.

(v) Stacking Ensemble which combines classifiers that use different learning algorithms.

(vi) Feature Subspace Ensemble creates each base classifier from a different sample from the feature space.

Bagging

Bagging (Breiman 1996) is a "bootstrap" (Efron and Tibshirani 1993) ensemble method that creates individual base classifiers by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, N examples - where N is the size of the original training set. In this technique many of the original examples from the training set may be repeated in the resulting training set while others may be left out.

Siyang et al. (2000) compared bagging and boosting using Naïve Bayes, Neural Networks and Decision Trees as the base classifiers using documents from Usernet postings. Their results showed that bagging increased the accuracy over using a single base classifier and performed better than the boosting ensemble.

Chawla et al. (2001) examined the different partitioning methods used to create a bagging ensemble and proposed a more intelligent way of partitioning into disjoint subsets using clustering. The intelligent partitioning method they use is called Fuzzy c-means (FCM). This algorithm begins with two clusters and clusters until the fuzzy membership values are stable. They found that the intelligent method of partitioning generally performs better than random partitioning. They also showed that bagging produced significant gain in accuracy over applying a single learner using the entire dataset. They suggest that this is due to bagging producing diverse classifiers from the data partitions.

In other research Zhou and Yu (2005) claim that bagging performs well for unstable classifiers such as neural networks or decision trees. They argue that perturbing the training set can cause significant changes in the construction of the unstable base classifier which increases Bagging accuracy. Bagging is not as effective with stable learners such as nearest neighbour classifiers, as perturbing the training set does not produce diverse base classifiers (Breiman 1996). Breiman (1996) suggests that bagging can be adapted for nearest neighbour algorithms by injecting randomness into the distance metrics. To do this they proposed a learning algorithm that randomly generates weights for calculating the Euclidean distance for each base classifier in the ensemble. They found that although this algorithm is simple, it can effectively improve the accuracy of nearest neighbour classifiers.

Boosting

The focus of boosting (Bauer and Kohavi 1999, Dong and Han 2005) is to produce a series of classifiers with each classifier's performance depending on the performance of the earlier classifiers in the series. The learner is applied to the training set as usual, except that each instance has an associated weight. All instances start with equal weights. Any misclassified training instances are given greater weight so that they are chosen more often than examples that were correctly predicted. The process is repeated until (a) the maximum number of classifiers is constructed or (b) there are no incorrectly classified training instances.

Dong and Han (2005) found boosting did not perform well when comparing it to bagging in the text classification domain. In their experiments they compared boosting and bagging with SVM as the base classifier on two training corpora namely Reuters-21578 (Lewis 1999) and 20-Newsgroup (Lang 1995). They also found that bagging and boosting are not effective when combining strong classifiers like SVM, with boosting always achieving the worse results.

Boosting has been applied in many domains outside of text classification, for instance Zhang and Rudnicky (2003) investigated Boosting and non-Boosting algorithms for acoustic model training on a real world continuous speech corpus. Using boosting for constructing ensembles of acoustic models, they found that there is no overall improvement between boosting and non-boosting methods. They suggest the reason for this is that the training data for speech recognition unavoidably contains a certain number of mislabels. In addition the training of acoustic models involves many stages, it might be better to focus on the characteristics of each stage rather than simply use a single overall algorithm for the whole training process.

Pal and Mather (2001) found that boosting increased the accuracy of their decision tree classifier. The classification problem involved the identification of six land cover types that cover the area of interest. They found in image classification that boosting increased the accuracy by 2% to 86.7%. They note that this is only a two percent increase in accuracy but they argue that small percentage increases in accuracy are difficult to generate when the overall classification accuracy exceeds 80%.

Round Robin

Round Robin classification (Fürnkranz 2003) also known as *pairwise* or *all vs. all* classification consists of an ensemble of binary classifiers. The basic idea is quite simple, namely to construct one classifier for each pair of classes. This technique transforms a n-class problem into $n(n-1)/2$ binary problems, one classifier for each pair of the n-classes. The binary classifier for problem (c_i, c_j) is trained with examples of class c_i and c_j, where examples of classes $C_k \neq c_i, c_j$ are ignored for this problem. The results of each binary classifier are combined depending on the classification returned. Each base classifier returns the class with the high score. These are then combined and the classification with the highest score is chosen as the ensemble classifier's classification.

Fürnkranz (2003) investigated the performance of round robin as a general ensemble technique. Furnkranz proposed a C5.0 Round Robin algorithm. Furnkranz found that due to the fewer training examples in each classifier in the ensemble, round robin is computationally no greater than bagging or boosting. Also the decision boundaries of each binary problem were found to be simpler than other ensemble techniques such as one vs. all ensembles where more complex functions are required to separate each class from all other classes. Furnkranz results showed that a round robin ensemble's performance is comparable to a single multi-category classifier, but it did not perform better than bagging and boosting, as the performance gain was not as large as the gain for bagging or boosting. This poor performance might be improved if the base classifiers had more examples to train from.

Grimaldi et al. (2002) used the round robin ensemble generation method with a simple k-NN classifier as the base classifier to classify music by genre. They compared it with a random classifier and a simple k-NN classifier. They found that Round Robin significantly outperforms the random classifier and the simple k-NN classifier used.

One Vs. All

One Vs. All (Rifkin and Klautau 2004) is a approach where an N class problem is transformed into N binary problems. N number of binary problems are constructed by using the examples of a particular class as the positive examples and using examples from all the other classes as the negative examples. A score is calculated for each base classifier (i.e. each class) and the classification with the highest score is chosen as the ensemble classifier's classification.

The results of each binary classifier are combined depending on the classification returned. For each classification a score is calculated ans the classification with the highest score is chosen as the ensemble classifiers classification.

Rifkin and Klautau (2004) argue that a simple *one vs. all* ensemble is as accurate as any other approach, assuming that the underlying base classifiers are well-tuned regularised classifiers such as SVM. Their results show that the *one vs. all* ensemble is as effective as round robin.

Stacking

Stacking (Dzeroski and Zenko 2004) is concerned with combining multiple classifiers generated by using different learning algorithms on a single dataset.

Dzeroski and Zenko (2004) evaluated several techniques for constructing ensembles of heterogeneous classifiers with stacking. The base classifiers they used were a decision tree learning algorithm C4.5 (Quinlan 1993), a k-NN algorithm, a NB algorithm (Rennie 2001a), a simple kernel density estimation algorithm and a multi-response linear regression algorithm (Ting and Witten 1999). Their results showed that stacking is no better than selecting the best classifier from the ensemble.

Sakkis et al. (2001) proposed a stacked generalization approach to anti-spam filtering. They used a memory-based classifier and a Naïve Bayes classifier in a two-member ensemble, in which another memory-based classifier presided over their stacked configuration. Their results show that stacking consistently improves the performance of anti-spam filtering over single classifiers. They attribute this to the presiding classifier's ability to choose the right ensemble member's decision when they disagree.

Feature Subspace Selection

Feature subspace selection (Grimaldi et al. 2003) is based on the idea of sub-sampling the feature space and training a classifier for each sub-space. The strength of this approach depends on having a variety of classifiers trained on different feature subsets sampled from the original space. In this method each ensemble member is trained on different feature-subsets of predefined dimension. Each feature-subset is drawn with replacement from the original set.

Grimaldi et al. (2003) evaluated a feature subspace based ensemble for the classification of music into different genres by comparing it to other techniques namely One vs. All, a simple k-NN classifier and Round Robin. They suggest that the feature selection can promote diversity among the ensemble members and thus improve their local specialization. They showed that the feature subspace selection based ensemble to be an effective ensemble technique which outperformed Round Robin, One vs. All and the single k-NN classifier.

2.4.2 Aggregation

This section describes the different methods of combining the results of the individual base classifiers into one result for the overall ensemble. These include Voting, Dynamic Classifier Selection and Meta-Classifiers.

Voting

There are typically two different types of voting used in combining the base classifiers results, majority voting and weighted majority voting. The simplest of these is majority voting, where the outputs of the k base classifiers are pooled together and the classification that reaches an overall majority is taken. Weighted majority voting is similar to majority voting but each base classifier has an associated weight which normally reflects the expected relative effectiveness of the classifier. The effectiveness of the classifiers can be determined on the basis of the individual classification accuracy on the weighted training sets.

Dynamic Classifier Selection

In this method of aggregation, the base classifier that yields the best performance on training examples that are most similar to the test example is selected and its classification is adopted by the ensemble. Giacinto and Roli (1999) using Naïve Bayes as their base classifiers, showed that selecting the best base classifier can outperform other aggregation approaches but it depends on the diversity of the base classifiers.

Meta-Classifier

In this approach for aggregating the outputs of the base classifiers, another classifier is used to combine the results. This meta-classifier reclassifies the results from the first layer of the ensemble.

In Siyang et al. (2000) research in text classification, they used different combinations of Naïve Bayes, Neural Networks and Decision Trees as their base classifiers and as their meta-classifier. Rather than solely relying on the output of the base classifiers they introduced a set of features that were a subset of the original feature space, which they suggest provides a low-dimensional abstraction. The top 40 features obtained from the feature selection for the base classifiers are used as the subset of features for the meta-classifier. They found that including these features increased the accuracy of the meta-classifier. A downside to using meta-classifiers is that it can increase the training and testing time as it adds an extra classifier to the number of base classifiers in an ensemble which adds to the computational complexity.

2.5 Conclusions

In this chapter text classification and its constituent stages have been outlined and explained (feature extraction and representation, dimensionality reduction and classification). This has shown that there are many ways of extracting and representing textual features, which can result in a very large feature space. There are also many ways of reducing this feature space (feature selection, feature transformation and wrapping), ranging from the computationally simple, such as document frequency threshold, to the more computationally complex, such as latent semantic indexing. This suggests that the feature extraction and representation and the dimensionality reduction processes can impact greatly the computational performance of a classifier.

This chapter has also shown that there are many ML techniques used in the classification process but described two of the more popular ML techniques, namely NB and SVM in more detail. Both have the ability to manage dimensionality but both being global, eager learners have the disadvantage of requiring a learning stage and can suffer from data interference when new instances are added to their training set.

Another ML approach reviewed in this chapter was case-based reasoning using the commonly used 4-R model to describe this lazy, local learner. This ML technique offers advantages over eager, global learners. These advantages are CBR does not generalise beyond the training data until it's given a new target case, the case-base is easily updatable and editable and its ability to handle diverse domains. It is our belief that with these advantages, CBR is well suited for the area of text classification.

This chapter then reviewed different similarity metrics, describing both feature based and featureless measures. Feature based similarity metrics calculate similarity by comparing the features of each instance while featureless similarity metrics calculate similarity by a general mathematical theory. Using featureless similarity measure requires no feature extraction and representation or dimensionality reduction processing which could impact the computational performance of a classifier.

This chapter finishes with a review of the various ensemble techniques used in ML. This review suggests that ensembles work well when the base classifiers are unstable such as decision trees or neural networks but suggests that stable classifier such as k-NN, could be used in ensembles if enough diversity is introduced. This diversity can be introduced through the ensemble techniques used in creating the ensemble namely the data partitioning. This suggests that the technique used to both partition the training data and to combine the base classifiers outputs can greatly increase or decrease the performance of the ensemble.

The next chapter presents the design of the case-base system used for the classification of text messages. It describe the feature extraction, case representation and dimensionality reduction used with the CBR system using a feature based similarity metric. It also describes the CBR system that uses a featureless similarity metric.

Chapter 3

System Design

As this thesis is related to the classification of general text messages which can contain varying lengths of text (i.e. long, short and/or a combination of both long and short text), this chapter first describes the four different corpora and the datasets derived from them (Section 3.1). It then describes the design of the CBR system used to classify text messages. This system was extended from the binary classifier, E-mail Classification Using Examples (ECUE) Delany et al. (2005), which is a binary CBR classifier used for spam email filtering, to a multi-category classifier, which can handle different types of text messages. The next section begins with a description of the CBR system using a feature based similarity metric. It describes the features that were extracted and their representations, followed by a description of the dimensionality reduction technique and the similarity metrics used. It continues with a description of the featureless similarity used and the design decisions made regarding the compression algorithm and file formats used. It then describes the case-base editing technique and the design of the classifier. This chapter finishes with a description of the ensemble techniques used.

3.1 Datasets

The datasets used to evaluate the different case-base configurations and different classifiers were obtained from four different text message corpora. Theses datasets contained examples of long and short text messages. Short text messages in this thesis are text messages that contain 500 characters or less (approx. the length of 3 SMS messages), while long text messages are very thing over 400 characters long. Two of which contain short text messages, one contains both short and long text messages and the other corpus contains long text messages. The different corpora are:

(i) Short Message Service (SMS), a collection of spam and legitimate SMS text messages.

(ii) Customer Comments corpus, a collection of customer comments from a large hotel chain about various aspects of their stay. This is a short text message corpus.

(iii) Helpdesk Emails corpus, a collection of emails recieved by a technical helpdesk which contain both long and short text messages.

(iv) Newsgroup 20 corpus, a collection of news threads documents which contain long text messages.

In this section, the different datasets that were derived from these four corpora are described and explained in detail. Each dataset underwent feature extraction (a feature being identified as a word, a letter or a statistical feature that was extracted during the feature extraction process, see section 3.2.1). The distribution and composition of each dataset is reported along with other characteristics such as average message length and the average feature frequency.

Short Message Service Datasets

The SMS corpus consists of two datasets with 100 legitimate and 100 spam messages in each. The legitimate SMS messages consist of personal and business text messages and the spam SMS messages contain promotional SMS messages and unwanted text alerts. While legitimate SMS messages are normally from personal correspondents and are normally short messages such as "Where are you?", spam SMS are normally from companies who are trying to offer some service or product such as "1000 Downloads 2 choose. Txt Sir to 80082 EUR3". The characteristics of the SMS datasets are shown in Table 3.1. These are the average message length and the distribution of the two different classes in the SMS corpus.

Table 3.1: SMS Dataset Characteristics

Class	Avg Msg Length	Distribution
Legitimate SMS	95	50%
Spam SMS	122	50%

Customer Comments Datasets

The customer comments corpus consists of over 5000 comments from guests visiting hotels that are part of a large hotel chain. The comments are graded in the range of 1 to 3 where

1 indicates high *satisfaction* with the service provided, 2 indicates low *satisfaction* and 3 are neither satisfactory or non satisfactory. The comments are also classified by the subject matter the customer is commenting about (i.e. comments about staff or the room they stayed in).

The comments themselves range from a few words e.g. "V good" to a detailed description of what a guest found good or bad e.g. "Enjoyed our stay in our family room immensely. Can't wait to come back." The customer comments dataset present a particular challenge for a text classifier as the difference between satisfactory and non satisfactory comments can be slight, for example "The room was good" and "The room was not good". This corpus was divided in three different ways, creating datasets with 2,3 and 4 classes.

For the binary datasets all customer comments with grade 1 were grouped into the class *satisfactory* and the comments with grade 2 and 3 were grouped into the class *unsatisfactory*. Four binary datasets were extracted each consisting of 500 satisfactory and 500 non satisfactory comments. These datasets are called the Satisfactory datasets. These binary datasets were used to test the CBR system on short text messages. The characteristics of the datasets are shown in table 3.2.

To create a dataset with 3 classes, the comments were divided on the grade of satisfaction given. 1000 customer comment were randomly selected and divided into three classes' grade 1, grade 2 and grade 3. This dataset is called the Grades dataset. To create a dataset with 4 classes, the comments were divided on the subject identifier. 1000 customer comments were randomly selected and divided into four classes, Food, Staff, Hotel and Room Equipment, this dataset is called the Subject dataset. The number of training samples used in each class follows the distribution of each class throughout the entire customer comments corpus. The characteristics of the Grades and Subject datasets are given in table 3.3 and table 3.4 respectively

Table 3.2: Customer Comments Satisfactory Datasets Characteristics

Class	Avg Msg Length	Distribution
Satisfactory	54	50%
Non-satisfactory	74	50%

Helpdesk Emails Dataset

The helpdesk corpus consist of 1000 emails sent to a technical support group within a large college. The emails are classified into four different categories. These categories are;

Table 3.3: Customer Comments Grade Dataset Characteristics

Class	Avg Msg Length	Distribution
Grade 1	44	47%
Grade 2	63	47%
Grade 3	50	6%

Table 3.4: Customer Comments Subject Dataset Characteristics

Class	Avg Msg Length	Distribution
Food	72	13%
Hotel	52	30%
Room Equip	62	17%
Staff	56	40%

(i) accessibility/service issues (i.e. creating email or user accounts and resetting passwords, changing access permissions)

(ii) automated/logging (i.e. automated emails from servers or automated response emails)

(iii) information email (i.e. email that conveys information e.g. telling the technicians to beware of a new virus circulating)

(iv) hardware/software issues (i.e. emails concerned with purchasing, installing, fixing or replacing software or hardware)

The challenge that this dataset presents is an email could contain text that could be classified as more than one class e.g. a request for both a new computer (a hardware issue) and to grant permissions to access a resource (accessibility issue). As we are dealing with single classification per instance, the most relevant classification is used to classify an instance. This was decided by manually choosing the most relevant classification for each instance in the training set.

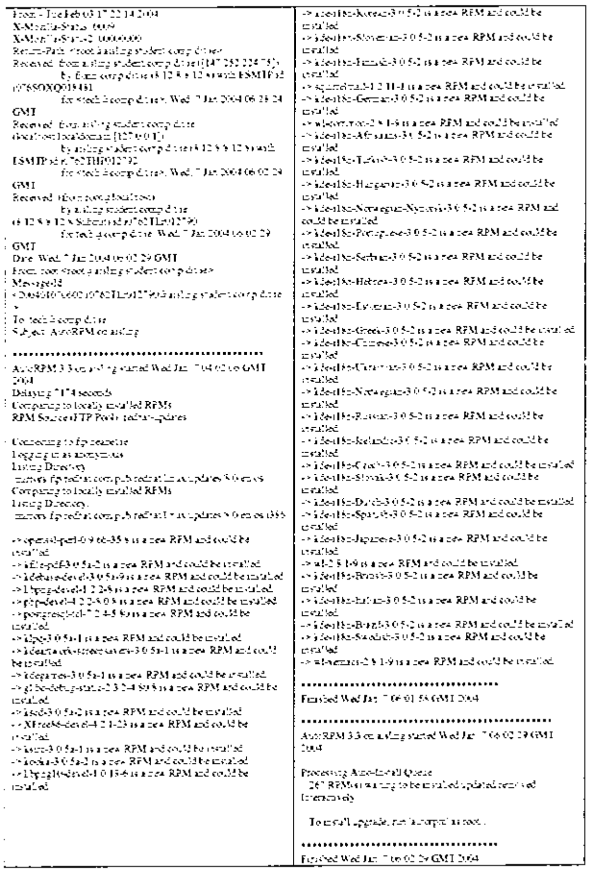
For this dataset an extra pre-classification process was added. From the emails header, the text in the TO, FROM, CC, BCC and SUBJECT tags was used, other text in the header is discarded. This was performed as not to add irrelevant features to the case-base (i.e. every instance would otherwise contain 'From' and other text that is present in every email header). Only the plain text from an emails main body is used. The characteristics of the helpdesk dataset are shown in table 3.5.

The average message lengths for the different classes in the Helpdesk dataset are reasonably large. These figures betray the categorization of this dataset as containing both

Table 3.5: Helpdesk Emails Dataset Class Characteristics

Class	Avg Msg Length	Distribution
accessibility/service issues	1035	34%
automated/logging	10988	13%
information email	1035	13%
hardware/software issues	1867	40%

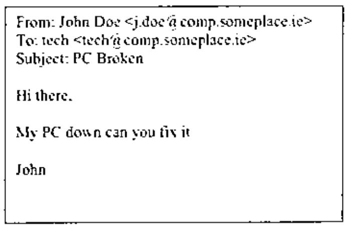
short and long text messages. This is due to some emails being extremely long while other are only a few sentences. This is illustrated in Figure 3.1 which shows two examples of a short email and a long email.



```
From - Tue Feb 03 17 22 14 2004
X-Mozilla-Status 0009
X-Mozilla-Status2 00000000
Return-Path <root@asling.student.comp.dcu.ie>
Received from asling.student.comp.dcu.ie ([147.252.224.75])
        by [illegible] comp.dcu.ie (8.12.8/8.12.8) with ESMTP id
i07[illegible]SOXQ019461
        for <tech@comp.dcu.ie>; Wed 7 Jan 2004 06 28 24
GMT
Received from asling.student.comp.dcu.ie
(localhost.localdomain [127.0.0.1])
        by asling.student.comp.dcu.ie (8.12.8/8.12.8) with
ESMTP id i0762THH012792
        for <tech@comp.dcu.ie>; Wed 7 Jan 2004 06 02 29
GMT
Received (from root@localhost)
        by asling.student.comp.dcu.ie
(8.12.8/8.12.8/Submit) id i0762TLr012790
        for tech@comp.dcu.ie; Wed 7 Jan 2004 06 02 29
GMT
Date Wed 7 Jan 2004 06 02 29 GMT
From root <root@asling.student.comp.dcu.ie>
Message-Id
<200401070602.i0762TLr012790@asling.student.comp.dcu.ie
>
To tech@comp.dcu.ie
Subject AutoRPM on asling

**********************************************
AutoRPM 3.3 on asling started Wed Jan 7 06 02 06 GMT
2004
Delaying 174 seconds
[illegible]
```

```
-> [illegible] is a new RPM and could be installed
[illegible]
**********************************
Finished Wed Jan 7 06 01 58 GMT 2004

**********************************************
AutoRPM 3.3 on asling started Wed Jan 7 06 02 29 GMT
2004

Processing Auto-Install Queue
  267 RPMs [illegible] to be installed/updated/removed
[illegible]

  To install/upgrade, run 'autorpm' as root.

**********************************
Finished Wed Jan 7 06 02 29 GMT 2004
```

Long Email



From: John Doe <j.doe@comp.someplace.ie>
To: tech <tech@comp.someplace.ie>
Subject: PC Broken

Hi there,

My PC down can you fix it

John

Short Email

Figure 3.1: Example of a long and short email

Newsgroup Datasets

The newsgroup datasets are derived from a corpus assembled by Ken Lang (Lang 1995). This dataset is a collection of approximately 20,000 newsgroup documents, partitioned evenly across 20 different newsgroups. The data is organized into 20 different classes, each corresponding to a different topic. This dataset has been used many times in text classification experiments, (Dong and Han 2005, Rennie 2001b, Schohn and Cohn 2000, Schapire and Singer 2000).

This dataset presents an interesting challenge as it contains many cross-posts to multiple newsgroups which can limit classification accuracy. Some of the topics are closely related (i.e. computer hardware and computer software) which gives the dataset two levels of granularity, groupings of similar classes under the same topic and the different classes from different topics. The topic groupings are Computer, Recreation, Science, Politics, Religion and Miscellaneous each of which have sub-classes. The structure of the Newsgroup corpus is shown in Fig 3.2.

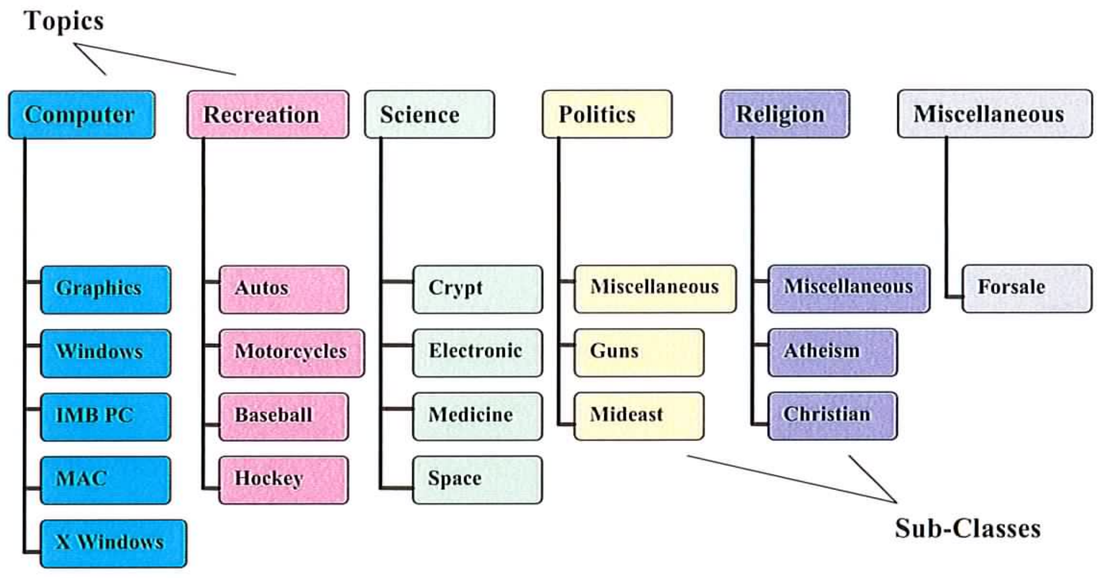


Figure 3.2: Newsgroup Dataset Structure

The corpus was divided into 5 datasets. The first dataset called NG-Topic consists of four out of the six topics namely Computers, Science, Politics and Recreation. For each topic 250 cases were randomly selected from the topics sub-classes. The average message length and average feature frequency for the NG-Topic dataset is given in Table 3.6.

The next dataset, NG-Close Relation, is a low granularity dataset consisting of four sub-classes from same topic. The classes in this dataset are closely related and therefore

Table 3.6: News Group Topics Dataset Class Characteristics

Class	Avg Msg Length	Distribution
Computers	4146	25%
Recreation	1526	25%
Science	2207	25%
Politics	4169	25%

harder to differentiate which presents a challenge for any classifier. The average case length and average feature frequency for the NG-Close Relation dataset is given in Table 3.7.

Table 3.7: News Group Close Relation Dataset Class Characteristics

Class	Avg Msg Length	Distribution
Windows	3717	25%
IBM PC	2100	25%
Mac	1758	25%
X Windows	1992	25%

For the next three datasets we selected eight sub-classes from the news group corpus, each with 250 randomly selected cases. Dataset NG-4 Classes Dataset is a dataset of four of these classes, NG-6 Classes Dataset includes six classes, while NG-8 Classes Dataset consists of all eight classes as illustrated in Fig 3.3. The characteristics of these datasets are given in Table 3.8.

Table 3.8: News Group 4, 6, 8 Classes Dataset Characteristics

Dataset	Class	Avg Msg Length
NG-4, 6, 8 Classes Dataset	Windows	2376
NG-4, 6, 8 Classes Dataset	Motorcycles	176
NG-4, 6, 8 Classes Dataset	Medicine	2191
NG-4, 6, 8 Classes Dataset	Christian	2558
NG-6, 8 Classes Dataset	Forsale	1263
NG-6, 8 Classes Dataset	Mideast	3254
NG-8 Classes Dataset	Atheism	2740
NG-8 Classes Dataset	Crypt	3673

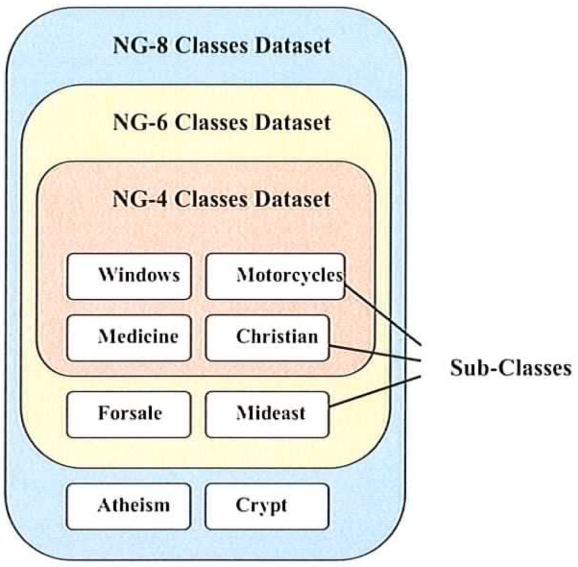


Figure 3.3: Newsgroup Dataset Structure

3.2 CBR using a Feature Based Similarity Metric

The design of a system for text message classification has to be adaptable and capable of representing an instance (i.e. a message) in a variety of different ways with respect to its feature types, feature representation and number of features that represent the actual instance. This section discusses the feature extraction process and the way in which each feature can be represented. It explains the dimensionality reduction process which selects the most predictive features.

3.2.1 Feature Extraction

Each instance was parsed and tokenised to identify the lexical features. No stop word removal, stemming or lemmatisation was performed on the messages before tokenisation. Three types of features were identified, these are

(i) Word features, a sequence of characters separated by white space.

(ii) Letter features or single character features.

(iii) Statistical features, e.g. the proportion of uppercase characters or the proportion of punctuation characters.

The statistical features that are used are:

(i) Proportion of uppercase letters

(ii) Proportion of lowercase letters

(iii) Proportion of punctuated character

(iv) Proportion of white spaces

(v) Lexical density, the proportion of the content (lexical) words over the total words

(vi) Average number of words per line

3.2.2 Feature Representation

In case-based reasoning each instance, in this situation each text message, is represented as a case. In this CBR system a case is represented as a vector of features (e.g. $c_i = (f_{i1}, f_{i2}...f_{in}, s)$ where f is a feature and s is the class).

To determine the value f_{ij}, for the feature f_i in case c_j, for numeric features, the normalised frequency of the feature is used, see Equation 3.1, where $freq_{i,j}$ is the number of times that feature f occurs in case c_j. This numeric feature representation is used for both word and letter features. The proportion calculated for the statistical features is used for their numeric feature representation, which is by definition between 0 and 1.

$$f_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}} \tag{3.1}$$

Binary feature representation for word features uses the existence rule i.e. if the feature exists in the case f_{ij} =1 otherwise $f_{ij} = 0$. For statistical and letter features the IG (Quinlan 1997) value is used, as calculated during the feature selection process (see section 3.2.3), to determine if f_{ij} is set to 1 or 0. This is determined by comparing the normalised frequency of the feature with the threshold value which returns the highest IG. If the normalised frequency is greater than the threshold value $f_{ij} = 1$ otherwise $f_{ij} = 0$.

3.2.3 Dimensionality Reduction

The feature extraction process produces a very large feature space, which is characteristic of text classification. To reduce this large feature space to a more manageable size, dimensionality reduction is employed. In this thesis IG (Quinlan 1997) is used in the dimensionality reduction process, as it was found to be effective in aggressively removing features without losing classification accuracy (Yang and Pedersen 1997). It was also shown to work well in a long text message domain of legitimate and spam emails (Delany et al. 2005).

3.3 CBR using a Featureless Based Similarity Metric

There are a number of parameters that need to be determined when using feature based similarity metrics, such as the feature types, their representations and the number of features to use that will give the lowest generalisation error. The process of determining these parameters can be time consuming. Conversely, a featureless similarity metric has the advantage of having no features. The featureless based similarity metric uses the compressed size of the instances to compute a similarity measure. Data compression is the process of encoding information using fewer bits than an un-encoded representation would use through use of specific encoding schemes (e.g. instead of using the word "compression" it can be encoded as "comp").

The Deflate algorithm is used as the data compression algorithm as it is independent of CPU type, operating system, file system and character set (Deutsch 1996). It is also used as the underlying compression algorithm in many different compression file formats, such as the Deflate and GZIP file formats and is not covered by patents (Deutsch 1996). This compression algorithm uses a combination of the LZ77 algorithm (Ziv and Lempel 1977) and Huffman coding (Huffman 1952).

The Deflate file format is used as it was found to perform better than the GZIP file format as shown in figure 3.4. A 10 fold cross-validation was performed on two datasets, one containing short text messages and the other containing long text messages. Each dataset contains 1000 instances and 4 classes with equal distribution. McNemar's test (Dietterich 1998) was used to determine whether any differences that existed were significant. The classifier used in this experiment was a k-NN with $k = 3$. The results show that the Deflate file format has a statistically significant lower generalisation error than the GZIP file format.

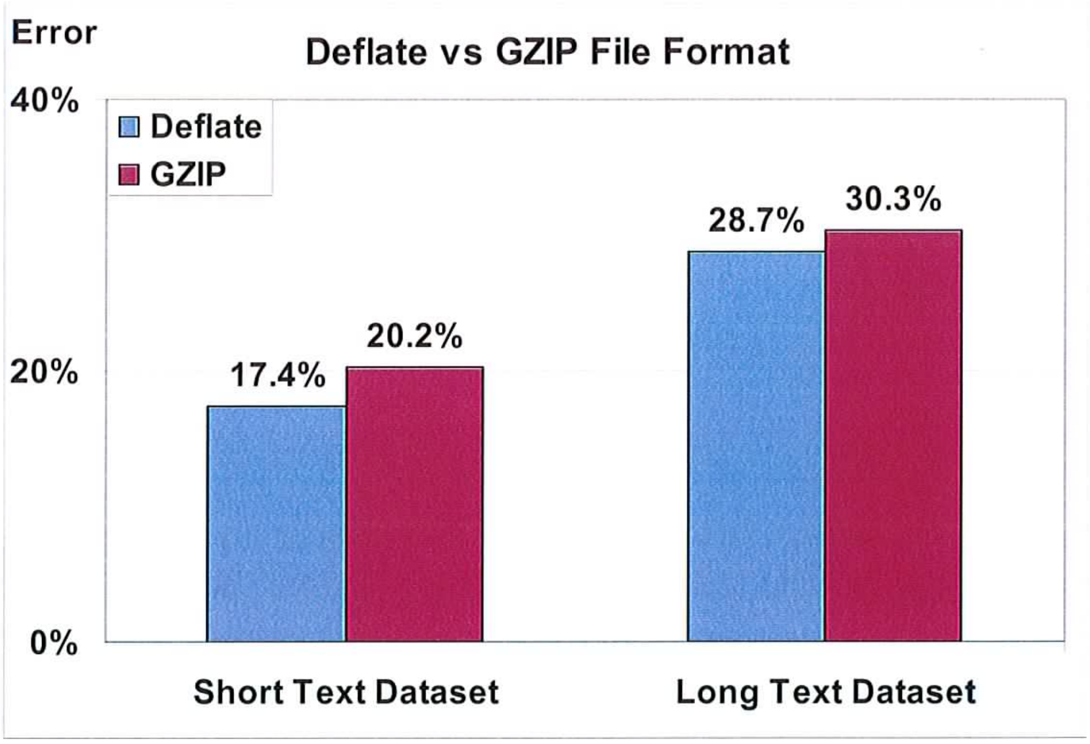


Figure 3.4: Results comparing Deflate and GZIP file format.

3.3.1 Featureless Similarity Metric

In this thesis two featureless similarity metrics are evaluated. The first was developed by Li et al. (2003) and is called normalised compression distance (NCD) (see section 2.3.2). The second is a variation of the Compression-based Dissimilarity Measure (CDM) (Keogh et al. 2004). The inverse of CDM is used as the similarity metric and is called Compression-based Similarity Measure (CSM). CSM is given in equation 3.2;

$$Sim(d_i, d_j) = \frac{1}{\left(1 - \left(\frac{C(d_i,d_j)}{C(d_i)+C(d_j)}\right)\right)} \tag{3.2}$$

where $C(d_i, d_j)$ denotes the compressed size of the concatenation of i and j , $C(d_i)$ denotes the compressed size of d_i , and $C(d_j)$ denotes the compressed size of d_j.

Figure 3.5 shows the results of comparing CSM with NCD. A 10 fold cross-validation was performed on two datasets, one containing short text messages (a random selection from the SMS and Customer Comments datasets) and the other containing long text messages (a random selection from the Newsgroup datasets) using a k-NN classifier with $k = 3$. Each dataset contains 1000 instances and 4 classes with equal distribution. McNemar's test (Dietterich 1998) was used to determine whether any differences that existed between CSM and NCD were significant. The results show that CSM outperforms NCD on both datasets with significant difference on the short text dataset. After this evaluation, CSM was chosen as the featureless similarity metric.

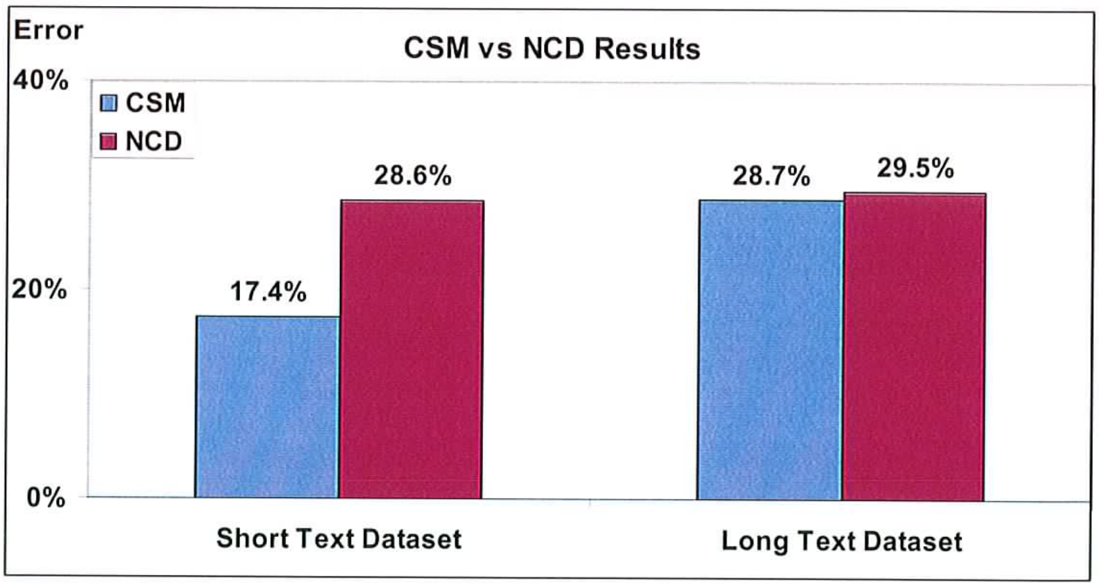


Figure 3.5: CSM vs NCD

For this similarity metric we considered weighting the voting to try and improve the classifiers performance. A number of mathematical operations were considered but squaring the similarity measure showed promise. The implementation of the weighted voting is given in equation 3.3

$$Sim(d_i, d_j) = CSM(d_i, d_j)^n \tag{3.3}$$

where n is the value of the weight. A 10 fold cross-validation was performed, varying the weight from 2 to 11, on two datasets, one containing short text messages and the other containing long text messages. The results in Figure 3.6 show that weighting the voting to the power of 7 produces the best performance for both short and long text message dataset. This suggests this weighting method reduces the possibility of a misclassification due to incorrectly classified instances having a higher weight than the correctly classified instances. This implementation of weighted voting with the featureless similarity metric, CSM^7 , will be used as it produced the best results.

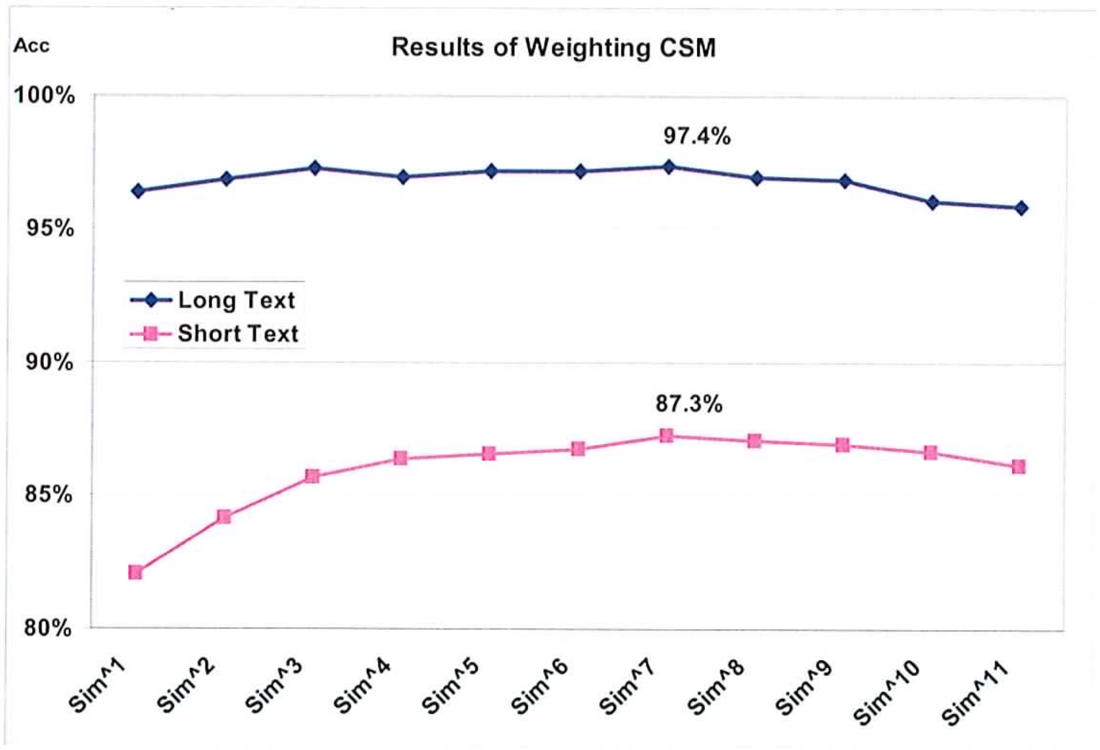


Figure 3.6: Results determining the appropriate weighted vote for CSM.

3.4 Case-base Editing

Case-base editing reduces the number of cases in the case-base. Textual case-bases can be extremely large due to the nature of text. They can become increasingly larger if new cases are being added to it, e.g. adding new training examples to the case-base when new cases present themselves. Editing a case-base can help maintain a case-base of reasonable size. Editing can also reduce the number of noisy cases or cases that do not contribute to the classification process, while trying to preserve or lower the classifiers generalisation error.

The editing technique used here is called Competence Based Editing (CBE) (Delany and Cunningham 2004) (see Section 2.2.6). This editing technique was found to lower the generalisation error on a case-base comprised of legitimate and spam email. CBE is evaluated to determine if it is appropriate for other text message domains.

3.5 The Classifier

The classifier used in the CBR system is the k-Nearest Neighbour (k-NN) classifier, which classifies a target case by analysing the training cases that are most similar to it. To compute the similarity between a target case d_t and a training case d_j a similarity measure is calculated. The feature base similarity measure used is given in equation 3.4.

$$Sim(d_t, d_j) = \sum_{f \in F} w_f \delta(d_{tf}, d_{jf}) \tag{3.4}$$

where F is the set of features and $\delta(d_{tf}, d_{jf})$ is described as

$$\delta(d_{tf}, d_{jf}) = \begin{cases} 1, & f \text{ binary and } d_{tf}, = d_{jf} \\ 0, & f \text{ binary and } d_{tf}, \neq d_{jf} \\ |d_{tf}, - d_{jf}|, & f \text{ numeric} \end{cases} \tag{3.5}$$

CSM is the featureless similarity measure used, as described in section 3.3.1

Once the k nearest neighbours are determined, a voting algorithm is implemented to determine the classification of the target case. The voting algorithm used is distance weighted voting where the k nearest neighbours vote on the classification of the target instance with votes weighted by their similarity to the query. The target case is assigned the classification with the highest score.

3.6 Ensemble Design

Classifier ensembles or classifier committees have been shown to produce better results than a single classifier (Dong and Han 2005, Grimaldi et al. 2003, Chawla et al. 2001). A number of ensemble techniques were evaluated. These include:

(i) One Vs All (OVA)

(ii) One Vs All with Bagging (OVA + Bagging)

(iii) One Vs All with Feature Subspace Selection (OVA + FSS)

(iv) One Vs All with Feature Subspace Selection and Bagging (OVA + FSS + Bagging)

(v) Round Robin (RR)

(vi) Round Robin with Bagging (RR + Bagging)

(vii) Round Robin with Feature Subspace Selection (RR + FSS)

(viii) Round Robin with Feature Subspace Selection and Bagging (RR + FSS + Bagging)

For each ensemble the base classifiers used were k-NN classifiers. The base classifiers results were combine using weighted majority voting. The votes were weighted by their similarity to the query. Figure 3.7 shows the results of a preliminary evaluation using 10 fold cross-validation comparing these ensemble techniques on a text dataset with 1000 instances and 4 classes. McNemar's test (Dietterich 1998) was used to determine whether any significant differences exist between the different ensemble techniques. The results show that One Vs All (OVA) and Round Robin (RR) produce the lowest generalisation error. These two ensemble techniques will be used to compare against a single k-NN classifier using both feature based and featureless similarity metrics.

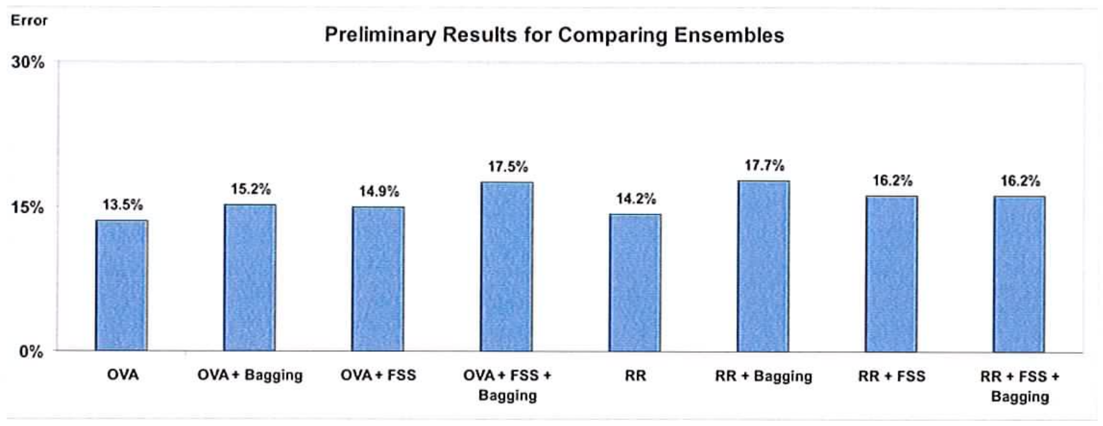


Figure 3.7: Preliminary results for comparing ensemble techniques.

3.7 Conclusions

This chapter discussed the design of the CBR system, using both feature based and featureless similarity metrics that will be used to classify text messages from different text domains. This chapter outlined the feature extraction and feature selection process for the feature based CBR system. It also described the various feature types and their representations the CBR system using a feature based similarity metric, is capable of representing.

This chapter continued to discuss the design decisions made for the CBR system using a featureless similarity metric, with respect to the compression algorithm and file format and the actual similarity metric used. It then discussed the case-base editing technique; CBE that will be used with this CBR system.

It then moved on to a discussion of the classifier (i.e. the k-NN algorithm), and finished with defining the ensemble techniques which will be used to compare against the single k-NN classifier.

The next chapter describes the experiments performed to evaluate the CBR system in various text domains with varying length text messages.

Chapter 4

Evaluation of CBR for Text Message Classification

The classification problem that is addressed in this thesis is the classification of text messages with varying message lengths and with a varying number of classes. The text messages can be of short length, such as SMS, or of long length such as discussion threads on the internet or even a combination of both short and long text such as emails or internet blogs. Text messages can occur in a number of different domains, such as email routing, online application forms or short message service (SMS). Classifying text messages in the various domains requires differentiating between varying numbers of classes, e.g. email routing for a technical helpdesk or the classification of internet blogs.

Because of these characteristics of text message classification, various evaluations were conducted to assess a case-based reasoning system's ability to classify both, and a mixture of, short and long text messages with multiple classes. Different datasets were used to cover different characteristics of text messages.

This chapter begins with the evaluation metrics and methods used in the evaluations. It continues with a description of the various evaluations performed on the CBR system using a feature based similarity metric. These include

(i) Evaluation to Determine Case Representation, which includes determining the appropriate number of features to use in the feature selection process, the feature types and feature representations.

(ii) Evaluation of a Case-base Editing technique, which includes determining if the case-base editing technique that was developed to reduce a case-base of spam and legitimate emails is appropriate for other text message domains.

It then describes the evaluation performed to determine the most appropriate similarity metric for the CBR system.

4.1 Evaluation Configuration

This section describes the justification for the evaluation metrics and the evaluation methods used for the different experiments.

4.1.1 Evaluation Metrics

For the various evaluations conducted, the classification error is reported. This is defined as the number of instances missclassified by the classifier and is given in Equation 4.1.

$$Err = \left(\frac{e}{N}\right)\left(\frac{100}{1}\right) \tag{4.1}$$

where e is the number of misclassified instances and N is the total number of instances.

For the SMS datasets, the rate of False Positives (legitimate SMS messages misclassified as spam) is also reported. As with other spam filtering such as email, a False Positive is more serious than a False Negative (a spam message misclassified as a legitimate message). A classifier used for spam filtering needs to minimise FP's, if not totally eradicate them completely. Also two classifiers could have the same error rate but have vastly different FP and FN rates. It is for this reason the FP rate is also reported. The FP rate is given in Equation 4.2

$$FPRate = \frac{\#FP}{\#FP + \#FN} \tag{4.2}$$

Where #FN is the number of False Negatives and #FP is the number of False Positives. In addition to these performance metrics, the resulting size of the edited case-base is also reported for the evaluation of the case-base editing technique

When the evaluation calls for the comparison of two classifiers, classifier A and classifier B, McNemar's test (Dietterich 1998) is used to calculate confidence levels to determine whether significant differences exist. It is appropriate for comparing classifiers because it does not assume independent samples (McNemar 1947). It also has some advantages over other performance measures (e.g. paired t-test), it has a lower Type I error (the probability of incorrectly detecting a difference when no difference exists) and has a better ability to detect a difference where one exists (Dietterich 1998). In order to compare two classifiers, the results from each are recorded on the same instance and four values are calculated. These are:

- n_{00}, the number of instances misclassified by both classifiers.
- n_{01}, the number of instances misclassified by classifier A but classified correctly by B.
- n_{10}, the number of instances classified correctly by classifier A but misclassified by B.
- n_{11}, the number of instances classified correctly by both classifiers.

The total number of test instances is $n = n_{00} + n_{11} + n_{01} + n_{10}$. If no difference exists between the two classifiers then $n_{10} = n_{01}$. In practice McNemar's test is carried out by a chi-square approximation with 1 degree of freedom. This statistic is given in Equation 4.3

$$\frac{(|n_{01} - n_{10}|)^2}{(n_{01} + n_{10})} \tag{4.3}$$

4.1.2 Evaluation Methods

The evaluation method used for each dataset was a 10 fold cross-validation, dividing the dataset into 10 stratified divisions or folds. In this method each fold in turn is used as a test dataset while the rest of the nine folds are considered to be the training dataset. Results are accumulated over all folds and reported for each dataset.

A case-base was built from each training dataset using the top N features ranked using IG (Quinlan 1997) (see section 3.2.3), where N is the number of features giving the lowest generalisation error. N was found by performing 10 fold cross validation varying the number of features until the error rate reach a minimum or reaches a plateau.

4.1.3 Experimental Setup

The machine the experiments where performed on was a 1.4GHz AMD Athlon with 512MB Ram memory, running Windows XP. The experiments were implemented in Java, version 1.4.2.

4.2 Evaluation of CBR for Text Message Classification

This section describes the evaluations performed on the CBR system for all the different datasets. The section begins with the evaluation to determine the most appropriate case representation for each of the different corpora. It then describes the evaluation to determine whether the case-base editing technique, Competence-Based Editing (CBE) lowers the generalisation error over an unedited case-base. CBE was found to work well for email spam filtering (Delany and Cunningham 2004), the evaluation in this section will determine whether it is suitable for other text domains. This section finishes with a comparison of feature based and featureless similarity metrics.

4.2.1 Evaluation to Determine Case Representation

This section describes the evaluations that were conducted to identify the most appropriate case representation for the four different corpora. This consists of determining the number of features to be used in the feature selection process, the combination of either word, statistical and/or single character features and the feature representation, either binary or numeric or a combination of both, that gives the lowest generalisation error for each of the four different corpora.

Determining the Number of Features

To determine the appropriate number of features to use in the feature selection process, the number of features was incremented from 10 until the accuracy peaks or reaches a plateau. As the best feature combination or feature representation has not been determined yet, all features (i.e. word, letters and statistical features) with binary representation are used. This combination is unlikely to give the best accuracy for all datasets, but should give the appropriate number of features required to give the lowest generalisation error. The results are shown in Table 4.1 and can be summarised as follows;

(i) The number of features required for the short text datasets (i.e. the SMS and Customer Comments datasets) range between 400 and 700 features. This is probably due to the small number of features in each message, because of which the classifier requires more features to differentiate between classes.

(ii) For the dataset which contains both short and long text messages, the Helpdesk dataset, the appropriate number of features is 600. This could be due to emails in the dataset which could have multiple classifications (i.e. can belong to more than one class). In such cases the classifier would require many features to help it differentiate between classes.

(iii) For the datasets which have long text messages, the Newsgroup datasets, the number of features were considerably lower than the other datasets. This may be due to the text messages using certain phrases and language that is unique to each class, such as "front side bus speed" or "motherboard" for computer related classes.

Table 4.1: Number of features for all datasets

Dataset(s)	No. of Classes	No. of Features
SMS Dataset 1-2	2	500
Customer Comments Satisfactory Dataset 1-4	2	700
Customer Comments Grades Dataset	3	500
Customer Comments Subject Dataset	4	400
Helpdesk Dataset	4	600
Newsgroup Topics Dataset	4	50
Newsgroup Close Relation	4	60
Newsgroup 4 Classes Dataset	4	90
Newsgroup 6 Classes Dataset	6	40
Newsgroup 8 Classes Dataset	8	40

Evaluation of Feature Types

The objective of this evaluation was to determine the combination of either word, statistical and/or single character features, that gives the lowest generalisation error. For each dataset the appropriate number of features, that were obtained earlier (see section 4.2.1), were used. For each dataset various combinations of features were accessed. The results are shown in Table 4.2 and can be summarised as follows:

(i) The feature types that gave the lowest generalisation error for the Helpdesk dataset were word and letter features. This result is interesting as none of the other datasets in this evaluation includes letter features. Although spam filtering another email domain also has letter features (Delany et al. 2004, 2005). This could be an idiosyncrasy of emails or because people use certain colloquialisms within emails such as 'LOL', 'ROLF' or 'c ya' which like the obfuscation used by spammers could explain why letters are so predictive in the email domain. Obfuscation is used by spammers to confuse email filters by including punctuation in the middle of words or by replacing certain letters such as 'i' with 1's or l's e.g. V.1:a.g.r:a. Spammers also tend to use a lot of uppercase characters e.g. I.N.V.E.S.T.M.E.N.T.

(ii) The features that gave the lowest generalisation error for the short text message and long text message datasets (i.e. SMS datasets, Customer Comments datasets and Newsgroup datasets) were word features and statistical features with no letter features. It is not surprising that letter features are not predictive for the SMS datasets as SMS spam is in its infancy and SMS spammers have not had to obfuscate their text messages to bypass filters yet, unlike email spammers who use obfuscation. For the Customer Comments datasets and the Newsgroup dataset the result is not that surprising as they both would contain structured English where letters would not necessarily be as predictive either.

Table 4.2: Feature Combinations for each Corpus

Msg Type	Corpus	Word Features	Letter Features	Statistical Features
Short	SMS	Yes	No	Yes
Short	Customer Comments	Yes	No	Yes
Short & Long	Helpdesk Dataset	Yes	Yes	No
Long	Newsgroup Datasets	Yes	No	Yes

Evaluation to Determine Feature Representation

The objective of this evaluation was to determine the feature representation that is appropriate for each of the datasets. The feature representation could be of either binary, numeric or a combination of both for the different feature types (i.e. words with binary representation and letters with numeric representation). The experiments were run using the already obtained appropriate number of features (see section 4.2.1) and the appropriate feature types (see section 4.2.1). For each dataset, the various combinations of feature representation were compared and assessed. Table 4.3 shows the different combinations of feature representation that were evaluated for the SMS, Customer Comments and Newsgroup corpora. Table 4.4 shows the different combinations of feature representation that were evaluated for the Helpdesk corpora.

Table 4.3: Feature Representation Combinations for the SMS, Customer Comments and Newsgroup corpora.

	Words	**Statistical**
Combination 1 (WS-BB)	Binary	Binary
Combination 2 (WS-NN)	Numeric	Numeric
Combination 3 (WS-BN)	Binary	Numeric
Combination 4 (WS-NB)	Numeric	Binary

Table 4.4: Feature Representation Combinations for the Helpdesk corpus

	Words	**Letters**
Combination 1 (WL-BB)	Binary	Binary
Combination 2 (WL-NN)	Numeric	Numeric
Combination 3 (WL-BN)	Binary	Numeric
Combination 4 (WL-NB)	Numeric	Binary

Figure 4.1 shows the results for the different datasets. The results can be summarised as follows

(i) There is no one feature representation appropriate for short text messages, as can been seen in Figure 4.1 where word and statistical features with binary representation (WS-BB) gives the lowest generalisation error for the SMS datasets while for the Customer Comments dataset, there is no overall feature representation that give the lowest generalisation error across the Satisfactory, Grades and Subject datasets. The feature representation that gives the lowest generalisation error, on average is numeric word and statistical features (WS-NN), although this is computationally expensive. If this was required in a real-time classification system, a feature representation of binary words and numeric statistical features (WS-BN) would be less computationally heavy. Also there is no significant difference between the two feature representations. It is for this reason binary words and numeric statistical features (WS-BN) was chosen as the preferable feature representation for the Customer Comments datasets.

(ii) For the dataset with a combination of short and long text messages, the Helpdesk dataset, the feature representation that gave the lowest generalisation error is binary words and numeric letter features (WL-BN). The difference is significant at the 95% level between binary words and numeric letter feature representation (WL-BN) and the other feature representation combinations except for an all numeric feature representation (WL-NN) where there is no significant difference although this feature representation is computationally heavy.

(iii) For the long text message datasets (i.e. the Newsgroup datasets) the feature representation combination that gives the best generalisation accuracy is an all numeric feature representation (WS-NN) (i.e. numeric word and statistical features). This feature representation combination is not significantly different to binary words and numeric statistical features for all the Newsgroup datasets. The differences are significant at the 95% or higher with all other feature representation combination in the 4, 6, 8 Classes datasets and the Topics dataset, the rest of the differences are not significant.

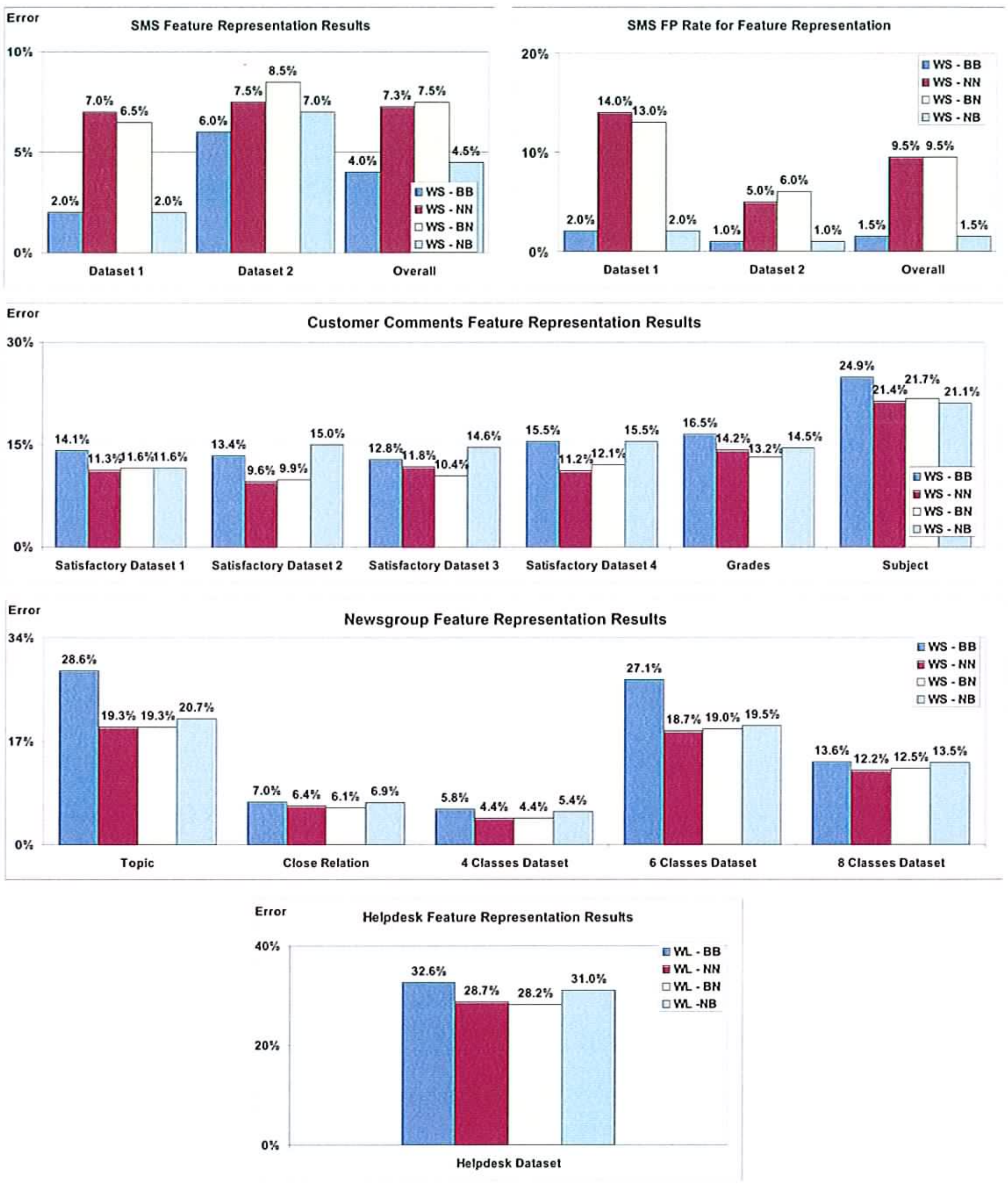


Figure 4.1: Results for the different feature representations

During this evaluation the value of k in the k-NN was also determined for each dataset, by performing a 10 fold cross validation varying the value of k. It was found that the value of k varied across the different type of datasets and the datasets themselves. The different values of k each of the datasets are shown in Table 4.5, which shows a brief summary of the results for the feature based case-base for all datasets.

Table 4.5: Results Summary for all Datasets

Corpus	Dataset	Msg Type	No. of Classes	Features	No. of Features	k
SMS	Dataset 1-2	Short	2	WS-BB	500	3
Customer Comments	Satisfactory 1-4	Short	2	WS-BN	700	7
	Grades	Short	3	WS-BN	500	7
	Subject	Short	4	WS-BN	400	11
Helpdesk	Helpdesk	Short& Long	4	WL-BN	600	3
Newsgroup	Topics	Long	4	WS-NN	50	3
	Close Relation	Long	4	WS-NN	60	9
	4 Classes	Long	4	WS-NN	90	7
	6 Classes	Long	6	WS-NN	40	3
	8 Classes	Long	8	WS-NN	40	3

4.2.2 Evaluation of Case-Base Editing Technique

This section describes the evaluation of the case-based editing technique called Competence Based Editing (CBE) (see Section 3.4) (Delany and Cunningham 2004) that was performed on all of the different datasets. Along with the error for each dataset being reported the reduced size of the case-base, as a percentage of the original case-base size, is also reported. Figure 4.2 shows the results for the evaluation of CBE. The conclusion from this is that CBE does not seem appropriate for editing a case-base outside of the spam email filtering domain. A more detailed summary of results is as follows;

(i) For the short text message datasets (i.e. SMS and Customer Comments datasets) the results show that the percentage error is higher for the edited case-base than the unedited case-base for the Customer Comments Datasets with all differences being significant at the 95% level or higher. While the SMS dataset shows little difference between edited and unedited case-bases this is probably due to the limited size of the SMS datasets. These results indicate that the editing technique CBE is not appropriate for short text message classification. One of the objectives of CBE was to conservatively reduce the size of the case-base (Delany and Cunningham 2004) by about 30%, but applying CBE to the Customer Comments and SMS datasets results in an overall reduction of between 50% and 65%. This suggests that the sparsity of the cases due to the short text content of the messages is not appropriate for the editing technique resulting in too many cases being removed.

(ii) The result for the Helpdesk dataset, which contains both short and long text messages, shows that the edited case-base performs better than the unedited case-base, but only slightly, although there is no significant difference. The interesting result for this dataset is that the edited case-base was reduced by approximately 25%, a conservative reduction which CBE was designed to do. This could be due to the feature representation containing letters.

(iii) For the long text message datasets, the Newsgroup datasets, the unedited case-base outperforms the edited case-base for all datasets except for the 6 Classes Dataset. For these datasets CBE reduces the case-base by an average of 41% and does not reduce the generalisation error. This could be due to the difference in case representations between these datasets and the spam filtering datasets.

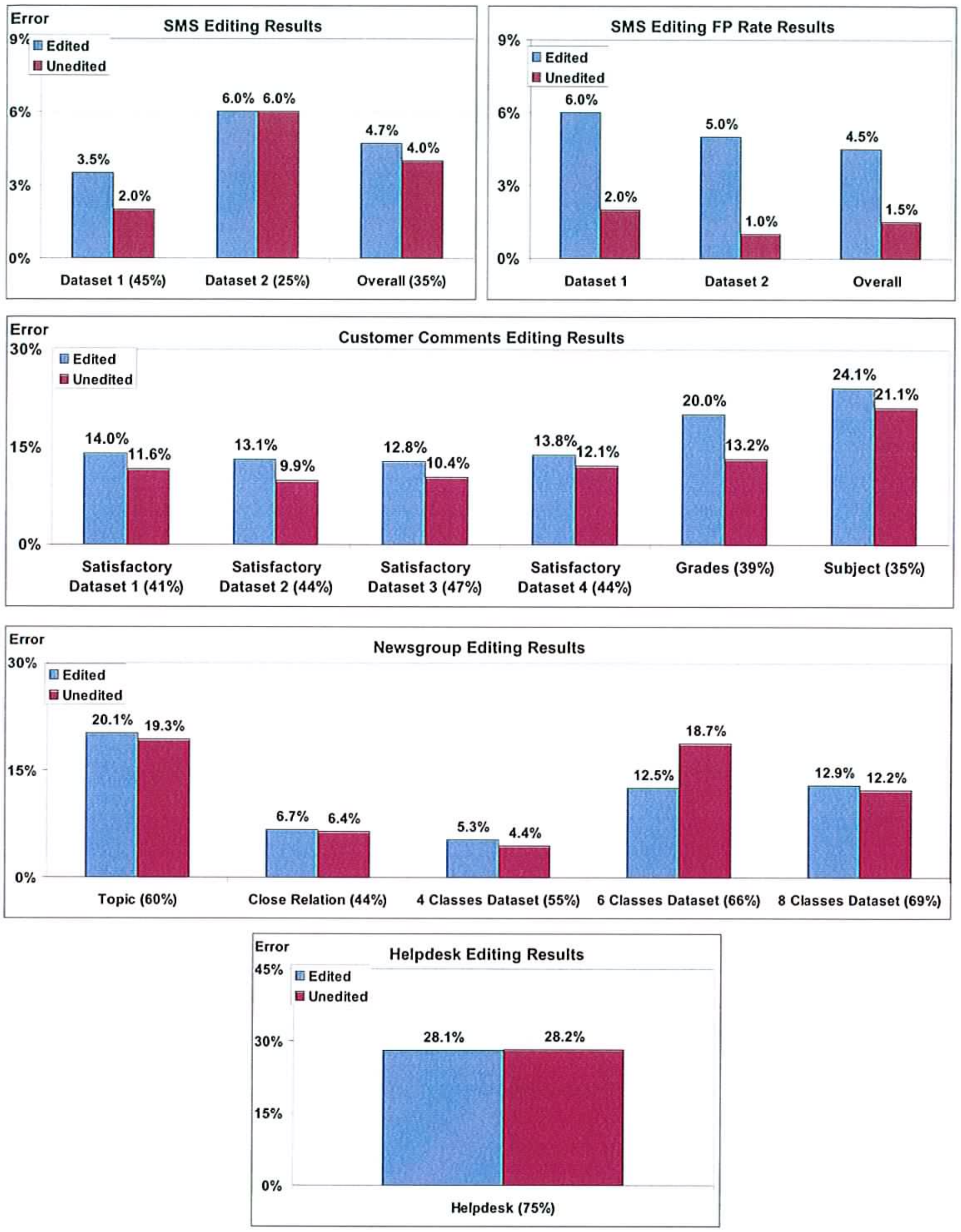


Figure 4.2: Evaluation Results for Case-base Editing

4.3 Comparing Feature Based and Featureless Similarity Metrics

A case-base using a feature based similarity metric extracts features from a case to compute the measure, which is then used to retrieve the most similar cases from the case-base. This method can be problematic as the feature types and their representations (e.g. binary or numeric) and the appropriate number of features has to be obtained. This can be time consuming and computationally expensive, although this method has been shown to be able to adapt to evolving text domains (Delany et al. 2004).

A case-base using a featureless similarity metric uses the compressed size of the cases in the case-base to compute a similarity metric, which is then used to retrieve the most similar cases from the case-base. This method of computing the similarity metric does not need the processes of finding the various parameters used in feature based similarity metrics. It can be however computationally expensive as each case is separately concatenated with every other case in the case-base and this is then compressed and used in the similarity metric. If the case-base is relatively large this can take a considerable amount of time.

Each method of computing the similarity between cases in the case-base has its pros and cons. To find which, if any, is superior, this section compares two case-bases, one using a feature based similarity metric, the other using CSM, the featureless similarity metric. The configurations that produced the lowest generalisation error are used for both the feature based similarity metric case-base (see section 4.2.1) and for CSM (see section 3.3). Figure 4.3 shows the results comparing the feature based and CSM and can be summarised as follows:

(i) Figure 4.3 shows that for short text messages, the case-base using the feature based similarity metric outperforms the case-base using CSM for most of the short text datasets, although the differences are not significant. The only exception is for the Customer Comments Subject dataset where the case-base using CSM has a lower generalisation error.

(ii) For the SMS false positive results show that CSM outperforms the case-base using the feature based similarity metric for both datasets. This result is interesting in that there are no false positives for either of the SMS datasets.

(iii) For the combination of short and long text messages dataset, the Helpdesk and the long text message datasets, the Newsgroup dataset, the case-base using CSM outperforms the case-base using the feature based similarity metric. There is only one exception, the Newsgroup Close Relation dataset where the case-base using the feature based similarity metric outperforms the case-base using CSM. This maybe due to the instances from the dataset being approximately the same size when compressed, as they have similar content, resulting in more misclassifications.

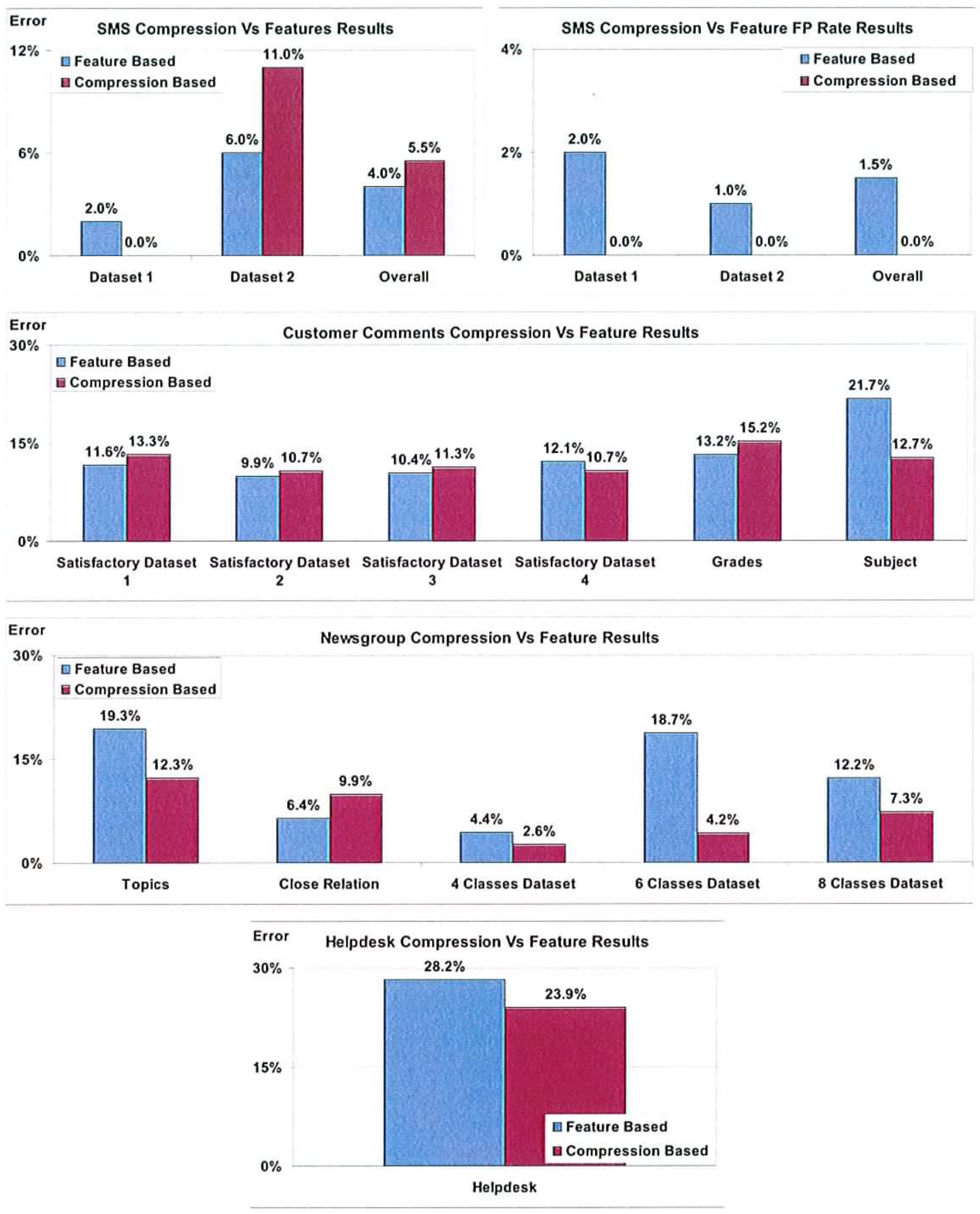


Figure 4.3: Results for comparing CSM with a feature based similarity metric

4.4 Conclusions

This chapter discussed the various evaluations performed on a case-base using both a feature based and featureless similarity metric using datasets representing different types of text message domains, namely short text messages, long text messages and a combination of both.

In this chapter, the most appropriate number of features, the appropriate feature types and the most appropriate feature representations were identified for datasets representing different types of text message domains for a case-base using a feature based similarity metric. We have shown that there is no "one size fits all" configuration for theses different text domains, and the case-base's configuration is dependant on the characteristics of the domain its being applied. This suggests that any case-base using a feature based similarity metric, which is used to classify text messages needs to be configurable in all these respects.

Our evaluation of the editing technique CBE, which was developed for maintaining a case-base of legitimate and spam email, is not appropriate for other text message domains. For the short text message datasets it seemed to remove too many cases resulting in the concept being learned being even weaker than it already was and for long text datasets it did not decrease the generalisation error.

We have shown that a case-base using a featureless similarity metric called CSM outperforms a case-base using a feature based similarity metric in text domains containing long text messages and a combination of long and short text messages. For short text message domains a case-base using CSM performs slightly worse than a case-base using a feature based similarity metric although the difference is not statistically significant.

Based on these results, a case-base using CSM, the featureless similarity metric, is superior to a case-base using a feature based similarity metric especially in text domains containing long text messages and a combination of long and short text messages and is comparable to feature based CBR in short text message domains.

The next chapter compares CBR using both feature based and featureless similarity metrics against other machine learning techniques used in text classification namely Naïve Bayes, Support Vector Machines and also ensembles of CBR classifiers.

Chapter 5

Comparing CBR with other Machine Learning Techniques

In chapter 4 the best configuration of the case-base was determined, both with a classifier using a feature based similarity metric and with a classifier using a featureless similarity metric. It is important to compare the case-based approach with other machine learning techniques on the same datasets, to see if CBR performs as well as some of the machine learning techniques of choice, Naïve Bayes and Support Vector Machines.

This chapter begins with a comparison of the case-based approach with different ensemble techniques. Ensemble techniques have been shown to improve on the results of single classifiers (Siyang et al. 2000, Zhou and Yu 2005, Fürnkranz 2003, Grimaldi et al. 2002, Dzeroski and Zenko 2004). In this comparison the One vs. All (OVA) and the Round Robin (RR) ensemble techniques are compared with the case-based approach. These were chosen as they gave the best results in preliminary evaluations (Section 3.6).

It finishes with a comparison between the case-based approach and two of the more popular machine learning techniques used for text classification, Naïve Bayes and Support Vector Machines. The case-based approach was evaluated using NB and SVM classifiers on each of the datasets.

5.1 Multi-Category Text Message Classification Using Ensembles of CBR Classifiers

Presented in this section is an evaluation to determine if ensembles of k-NN classifiers perform better than a single k-NN classifier on multi-category datasets. The different ensemble techniques that were used for this evaluation were;

(i) One Vs. All (OVA)

(ii) Round Robin (RR)

These ensemble techniques were chosen as they were found to give the best results in preliminary evaluations of different ensemble techniques (Section 3.6). The evaluation method used for this evaluation was a 10 fold cross-validation. Results are accumulated over all folds and reported for each dataset. McNemar's test was used to determine if any significant difference existed between the classifiers.

This evaluation compared both a k-NN with CSM and k-NN with a feature based similarity metric with the ensemble classifiers. Four ensemble classifiers where created using both k-NNs with CSM and k-NNs with feature based similarity metrics as their base classifiers. The ensemble classifiers are

(i) OVA using k-NN with a feature based similarity metric as the base classifiers.

(ii) RR using k-NN with a feature based similarity metric as the base classifiers.

(iii) OVA using k-NN with the CSM similarity metric as the base classifiers.

(iv) RR using k-NN with the CSM similarity metric as the base classifiers.

For this evaluation the datasets used are the Customer Comments Grades and Subject datasets, the Helpdesk dataset and the Newsgroup Datasets. Figure 5.1 show the results of this evaluation. It shows the error rate for the single k-NNs and the four different ensembles. It shows that the ensemble techniques do not lower the generalisation error when compared with the single case-base using the CSM similarity metric. The results can summarised as follows;

(i) The single k-NN using the CSM similarity metric outperforms the four different ensembles in all the datasets except for the Newsgroup Close Relation datasets.

(ii) The differences between the ensemble techniques and the single k-NN using CSM are significant for all datasets except for the Customer Comments Grades datasets

This result is surprising as ensembles normally improve on the results of a single classifier. This result could be due to using stable classifiers (k-NNs) for the base classifiers in the ensembles or that there was insufficient diversity introduced into the training data through the partitioning techniques used.

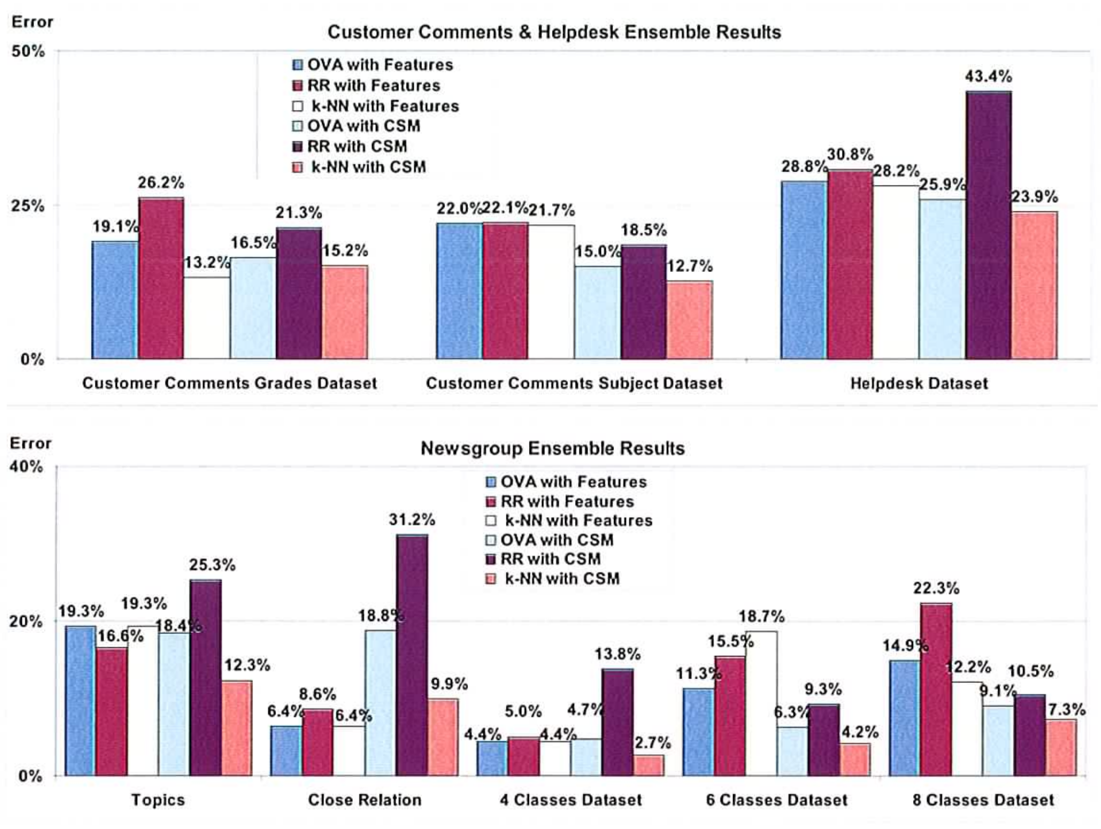


Figure 5.1: Results for comparing the different ensemble techniques with a single case-base using the CSM similarity metric and a case-base using a feature based similarity metric.

To rule this out an evaluation was performed comparing CSM and a technique that is designed to introduce diversity into the training data, Feature Subspace Selection (FSS) (see section 2.4.1). Due to computational limitations only the Customer Comments datasets and the Helpdesk dataset are used in this evaluation. Figure 5.2 shows the results of this evaluation comparing a single k-NN using CSM with a One Vs All ensemble using FSS. The results can be summarised as follows;

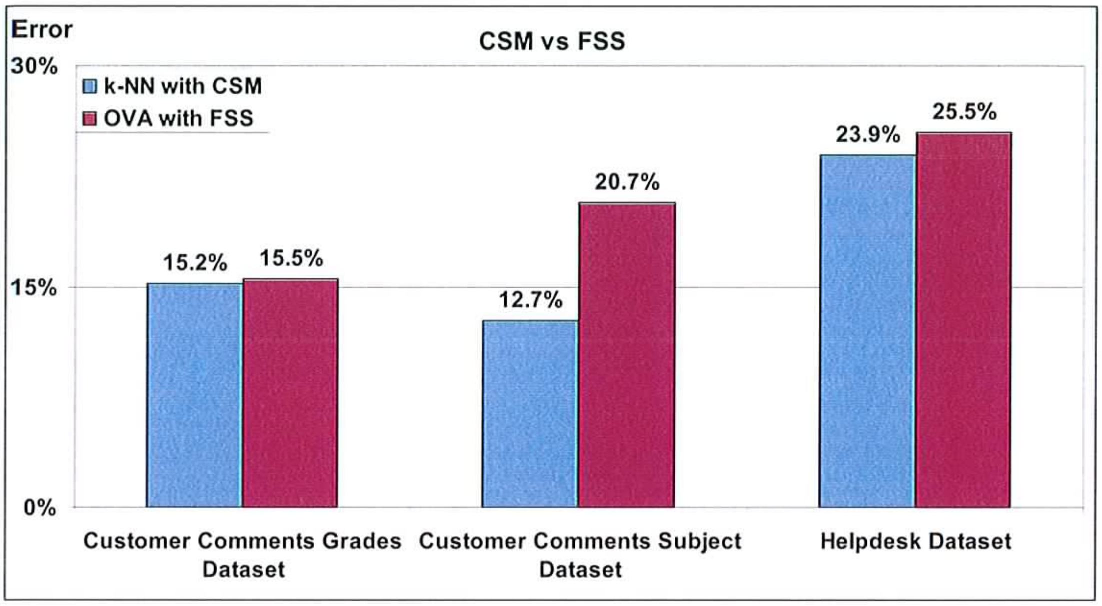


Figure 5.2: Results for comparing CSM and a One Vs All ensemble using FSS

(i) The single k-NN using the CSM similarity metric outperforms the One Vs All ensemble using FSS for all datasets. The difference between k-NN using the CSM and the One Vs All ensemble using FSS is significant at the 95% level or higher for the Customer Comments Subject dataset and Helpdesk dataset. This could be due to FSS overfitting on the training data.

5.2 Comparing CBR with NB and SVM

There are many machine learning techniques used for text classification and two of the most popular at the moment are Naïve Bayes (NB) (Mitchell 1997) and Support Vector Machines (SVMs) (Christianini and Shawe-Taylor 2000). It is important to compare the best configuration of the case-base with these two popular machine learning techniques. NB is currently one of the most popular classifiers in such applications as spam filtering (Androutsopoulos et al. 2000, Sahami et al. 1998) and in information retrieval (Ghani 2001, Faloutsos and Oard 1995), while there has been considerable amount of research into SVM for the purpose of text classification (Christianini and Shawe-Taylor 2000, Tang 2001, Joachims 1999, Platt 1999, Huang 2003).

As NB and SVM are both feature based classifiers (i.e. they used features extracted from each instance), the configuration of the case-base using the feature base similarity metric that gave the lowest generalisation error for each dataset is used. Table 5.1 gives a summary of the configuration for the different datasets.

Table 5.1: Results Summary for all Datasets

Corpus	Dataset	Msg Type	No. of Classes	Features	No. of Features	k
SMS	Dataset 1-2	Short	2	WS-BB	500	3
Customer Comments	Satisfactory 1-4	Short	2	WS-BN	700	7
	Grades	Short	3	WS-BN	500	7
	Subject	Short	4	WS-BN	400	11
Helpdesk	Helpdesk	Short& Long	4	WL-BN	600	3
Newsgroup	Topics	Long	4	WS-NN	50	3
	Close Relation	Long	4	WS-NN	60	9
	4 Classes	Long	4	WS-NN	90	7
	6 Classes	Long	6	WS-NN	40	3
	8 Classes	Long	8	WS-NN	40	3

The implementation of the NB classifier is given in equation 2.9, which incorporates a Laplace correction, as described in section 2.1.3. The implementation used for the SVM is WEKA's version of a SVM called Sequential Minimal Optimization (SMO) (Witten and Frank 2005, Garner 1995). WEKA stands for Waikato Environment for Knowledge Analysis which was developed at the University of Waikato in New Zealand and is publicly available online at http://www.cs.waikato.ac.nz/ml/weka. It implements John C. Platt's sequential minimal optimization algorithm (Platt 1999) for training a support vector classifier using polynomial. This implementation replaces missing values, transforms nominal attributes into binary attributes and normalises all attributes by default. For multi-class problems, it uses the round robin ensemble technique. Figure 5.3 shows the results of this evaluation. The results can be summarised as follows;

(i) For the long text messages and the combination of both short and long text datasets (i.e. the Newsgroup and Helpdesk datasets), the k-NN using CSM outperforms both NB and SVM except for the Newsgroup Close Relation Dataset. While there is a significant difference between the k-NN using CSM and SVM for the Newsgroup datasets, there is none for the Helpdesk dataset. The poor performance of the SVM on the Newsgroup dataset is probably due to the low number of features used. SVMs normally performs well with high dimensionality datasets, but for the Newsgroup datasets the number of features is extremely low, between 40 and 90 features, which could explain why SVM is having difficulty differentiating between the different classes. The difference between the k-NN using CSM and NB is significant for the 6 and 8 Classes datasets and the Helpdesk dataset.

(ii) For the short text message datasets the k-NN using CSM does not perform as well as the NB classifier or the SVM classifier for all of the datasets except for the SMS Dataset 1. This result is probably due to the small size of each instance in these datasets (i.e. the low number of features in each instance), although CSM does have the lowest FP rate when compared with the other classifiers. There is a significant difference between both k-NN's and NB for all short text message datasets except for the Customer Comments Subject dataset and SMS Dataset 1. The difference between the k-NN using CSM and the SVM is only significant for the Customer Comments Satisfactory dataset 1 and 2 and the Grades datasets. For the k-NN using the feature based similarity metric and the SVM is significant for all datasets except Customer Comments Satisfactory dataset 3, Grades dataset and SMS dataset

2.

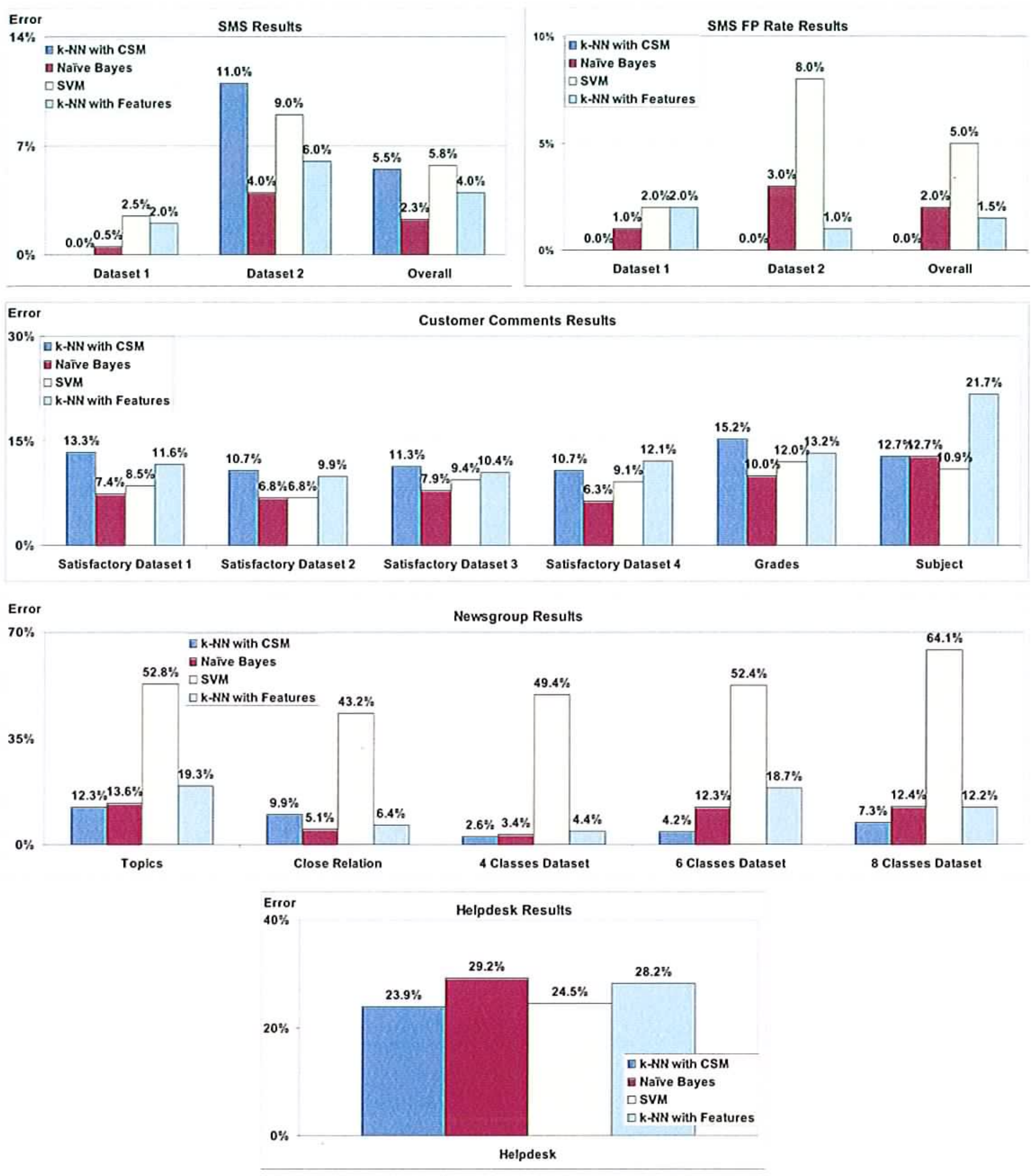


Figure 5.3: Results for comparing a single case-base using the CSM similarity metric with NB and SVM.

5.3 Conclusions

In this chapter two case-bases, one with a k-NN using the CSM similarity metric and the other with a k-NN using a feature based similarity metric were compared with various ensemble techniques and two other popular machine learning techniques, NB and SVM.

The comparison of the single k-NN with the OVA and RR ensemble techniques, showed that the case-based approach with a k-NN using the CSM similarity metric performs better than the different ensemble techniques. This could be due to the base classifiers being used are stable classifiers and do not introduce enough diversity into the training data. We then compared an OVA ensemble using Feature Subspace Selection with a k-NN using CSM. The results showed that CSM outperformed the ensemble, this could be due to FSS ensemble overfitting the training data.

We have also shown that a case-base using the CSM similarity metric optimised with respect to compression algorithm, file format and weighted voting, can outperform two more popular machine learning techniques, namely NB and SVM which were not optimised, on datasets containing long text messages. We also showed that the case-base using a k-NN with the CSM similarity metric does not perform as well as the k-NN using the feature based similarity metric, NB or SVM on datasets containing short text messages.

Chapter 6

Conclusion

A considerable amount of effort and research has been done on the automation of the classification of text messages. Difficulty arises in that text messages can be diverse (e.g. the different types of spam email), they may contain short, long or a combination of both short and long text (e.g. email, message board posts, SMS messages, internet blogs) and can be constantly changing over time (e.g. concept drift in spam email or the changing language used in SMS messaging).

The challenge is to find a machine learning approach that has the ability to adapt to the diversity and weak or strong concepts of different text domains and that can tackle the incremental learning problem of the domains constantly changing. The focus of this thesis was the application of a case-based approach using a k-NN classifier for the classification of text messages. CBR offers distinct advantages over other eager learners such as NB or SVM. It has the advantage of being a lazy, local learner which can perform incremental learning without the need of a separate learning process and can handle diverse domains.

We compared two case-based approaches, one using a k-NN classifier with a featureless similarity metric and the other using a k-NN classifier with a feature based similarity metric. We evaluated both case-based approaches on several different datasets, which represents different types of text messages (i.e. email, SMS and message board posts).

For the k-NN with a feature based similarity metric we determined the best feature types, feature representations and the number of features to represent a case. We have shown that the configuration of a case-based with regard to the types and combinations of features, feature representation and the number of feature used in the dimensionality reduction stage depends on the text message domain. Each of the different types of datasets used in this thesis used a different configuration of feature types and feature representations. They also required different number of features to be used in their case representation to produce the best classification accuracy. We also found that the inclusion of letter features in the classification of email datasets improves the classification accuracy, as the helpdesk dataset required letter features and previous research in spam filtering used letter features (Delany 2006, Delany et al. 2005).

Next we evaluated a case-based editing technique that was developed for editing a case-base of legitimate and spam emails. The editing technique was found to increase generalisation error for all of the datasets except the email dataset, the Helpdesk dataset. This editing technique seems to be specifically for the email domain.

We developed a featureless similarity metric called Compression-based Similarity Measure (CSM) based on Keogh et al. (2004)'s Compression-based Dissimilarity Measure (CDM). Using CSM has a number of advantages over a feature based similarity metric, there is no feature extraction and representation or dimensionality reduction process. It works on the raw text of the case. We compared both case-based approaches and have empirically shown that a case-base using a k-NN with CSM performs better than a case-base using a k-NN with a feature based similarity metric on datasets that contain long text messages. Although CSM performed well with datasets that contain long text messages, we found that a case-based approach using a k-NN with CSM performs slightly worse than a case-based approach using a k-NN with a feature based similarity metric on datasets that contain short text messages although there is no statistically significant difference. This is probably due to the case containing very little raw text.

As ensemble techniques normally improve the performance of a single classifier, we evaluated our case-based approach with two different ensemble techniques, one vs. all and round robin. The results show that these different ensemble techniques do not perform as well as a single k-NN classifier using either CSM or feature based similarity metric.

We also compared our case-based approach against two popular machine learning techniques, Naïve Bayes and Support Vector Machine, on all of the datasets. We found that a case-base using CSM can, under certain circumstances perform better than Naïve Bayes and Support Vector Machine on datasets that contain long text messages except when the classes are extremely closely related. We do note that both the NB and the SVM where not optimised to the same extent as our CBR system. For the CBR system the feature types, feature representation and number of feature were optimised. When the dataset contains classes that are closely related, the resultant compressed cases are of similar size and therefore the similarity measures produced by CSM are extremely close, which causes a loss in accuracy. On a side note we also found that a SVM classifier produces poor results when there is limited number of features in the case representation, whereas CBR and NB can handle the low dimensionality. This is evident with the Newsgroup datasets where the number of features in the case representation range between 40 and 90 features, and the SVM generalisation error is between 43% and 65%, whereas CBR and NB generalisation error range between 2% and 15%.

There is no classification technique that would ensure 100% accuracy, but CBR does provide certain advantages over other techniques and can perform as well as or better than other widely used machine learning techniques. We believe that CBR is a versatile machine learning paradigm that can be applied to many real world applications and can contribute to the task of organising the vast amounts of information available.

In this thesis we have compared different case representation and different similarity metrics for different types of text datasets. We have shown that a text classification system has to be configurable to achieve the best performance with a certain dataset type. Our future work in this area is to extend the current system to automatically detect and configure itself with the best case representation and similarity metric. We propose that an analysis of the dataset would herald information that could be used to select the best configuration instead of performing numerous cross-validation experiments.

Bibliography

Aamodt, A. and Plaza, E.: 1994, Case-based reasoning : Foundational issues, methodological variations, and system approaches. AI Communications, 7(1), March 1994.
URL: *citeseer.ist.psu.edu/252387.html*

Aas, K. and Eikvil, L.: 1999, Text categorisation: A survey. Technical report, Norwegian Computing Center, June 1999.
URL: *citeseer.ist.psu.edu/aas99text.html*

Alpaydin, E.: 2004, *C4.5 Programs for Machine Learning*, The MIT Press, Cambridge Massachusette, London, England.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V. and Spyropoulos, C. D.: 2000, An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages, *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, pp. 160–167.

Ashley, K. D. and Aleven, V.: 1991, Toward an intelligent tutoring system for teaching law students to argue with cases, *ICAIL '91: Proceedings of the 3rd international conference on Artificial intelligence and law*, ACM Press, New York, NY, USA, pp. 42–52.

Atkeson, C. G., Moore, A. W. and Schaal, S.: 1997, Locally weighted learning, *Artificial Intelligence Review* 11(1-5), 11–73.
URL: *citeseer.ist.psu.edu/atkeson96locally.html*

Attardi, G., Gullí, A. and Sebastiani, F.: 1999, Automatic Web page categorization by link and context analysis, *in* C. Hutchison and G. Lanzarone (eds), *Proceedings of THAI-99, European Symposium on Telematics, Hypermedia and Artificial Intelligence*, Varese, IT, pp. 105–119.
URL: *citeseer.ist.psu.edu/attardi99automatic.html*

Bauer, E. and Kohavi, R.: 1999, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, *Machine Learning* **36**(1-2), 105–139.
URL: *citeseer.ist.psu.edu/bauer99empirical.html*

Breiman, L.: 1996, Bagging predictors, *Machine Learning* **24**(2), 123–140.
URL: *citeseer.ist.psu.edu/breiman96bagging.html*

Brighton, H. and Mellish, C.: 2002, Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery 6 (2002) 153-172.
URL: *citeseer.ist.psu.edu/brighton02advances.html*

Bruninghaus, S. and Ashley, K. D.: 2001, The role of information extraction for textual CBR, *Case-based reasoning research and development*, Springer Verlag, pp. 74–89.
URL: *citeseer.ist.psu.edu/452192.html*

Calvo, R. A. and Ceccatto, H. A.: 2000, Intelligent document classification. Intelligent Data Analysis, 4(5), 2000.
URL: *citeseer.ist.psu.edu/calvo00intelligent.html*

Caropreso, M. F., Matwin, S. and Sebastiani, F.: 2001, A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization, *in* A. G. Chin (ed.), *Text Databases and Document Management: Theory and Practice*, Idea Group Publishing, Hershey, US, pp. 78–102.
URL: *citeseer.ist.psu.edu/379242.html*

Chawla, N., Eschrich, S. and Hall, L. O.: 2001, Creating ensembles of classifiers, *Proceedings of ICDM*, pp. 580–581.
URL: *citeseer.ist.psu.edu/chawla01creating.html*

Christianini, N. and Shawe-Taylor, J.: 2000, An introduction to support vector machines and other kernel-based learning methods, *Robotica* **18**(6), 687–689.

Cohen, W. W. and Singer, Y.: 1996, Context-sensitive learning methods for text categorization, *in* H.-P. Frei, D. Harman, P. Schäuble and R. Wilkinson (eds), *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, ACM Press, New York, US, Zürich, CH, pp. 307–315.
URL: *citeseer.ist.psu.edu/cohen96contextsensitive.html*

Coyle, L., Hayes, C. and Cunningham, P.: 2002, Representing cases for cbr in xml, *proceedings of 7th UKCBR Workshop, Peterhouse, Cambridge, UK.*, Springer.
URL: *citeseer.ist.psu.edu/coyle02representing.html*

Cunningham, P. and Dahyot, R.: 2004, Review of machine learning techniques for processing multimedia content, *Proceedings of The 22nd International Conference on Machine Learning.*

Delany, S., Cunningham, P. and Coyle, L.: 2004, An assessment of case-based reasoning for spam filtering, *Proceedings. of 15th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 9–18.

Delany, S., Cunningham, P., Tsymbal, A. and Coyle, L.: 2005, A case-based technique for tracking concept drift in spam filtering, *Knowledge-Based Systems* 18(4–5), 187–195.

Delany, S. J.: 2006, *Using Case-Based Reasoning for Spam Filtering*, PhD thesis.

Delany, S. J. and Cunningham, P.: 2004, An analysis of case-based editing in a spam filtering system, *in* P. Funk and P. González-Calero (eds), *Proceedings of 7th European Conference on Case-Based Reasoning (ECCBR 2004)*, Vol. 3155 of *LNAI*, Springer, pp. 128–141.

Deutsch, L. P.: 1996, Deflate compressed data format specification.
URL: *ftp://ftp.uu.net/pub/archiving/zip/doc/*

Dewdney, N., VanEss-Dykema, C. and MacMillan, R.: 2001, The form is the substance: classification of genres in text, *Proceedings of the workshop on Human Language Technology and Knowledge Management*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 1–8.

Dietterich, T. G.: 1998, Approximate statistical test for comparing supervised classification learning algorithms, *Neural Computation* 10(7), 1895–1923.
URL: *citeseer.ist.psu.edu/dietterich98approximate.html*

Dong, Y.-S. and Han, K.-S.: 2005, Boosting svm classifiers by ensemble, *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, ACM Press, New York, NY, USA, pp. 1072–1073.

Dumais, S., Platt, J., Heckerman, D. and Sahami, M.: 1998, Inductive learning algorithms and representations for text categorization, *CIKM '98: Proceedings of the seventh in-*

ternational conference on Information and knowledge management, ACM Press, New York, NY, USA, pp. 148–155.

Dzeroski, S. and Zenko, B.: 2004, Is combining classifiers with stacking better than selecting the best one?, *Machine Learning* **54**(3), 255–273.

Efron, B. and Tibshirani, R. J.: 1993, *An introduction to the Bootstrap.*, Vol. 57 of *Monographs on Statistics and Applied Probability*, Chapman and Hall.

Faloutsos, C. and Oard, D. W.: 1995, A survey of information retrieval and filtering methods, *Technical report*, College Park, MD, USA.

Fox, C.: 1990, A stop list for general text, *SIGIR Forum* **24**(1-2), 19–21.

Fox, C.: 1992, Lexical analysis and stoplists, pp. 102–130.

Frakes, W. B.: 1992, Stemming algorithms, pp. 131–160.

Fred J. Damerau, Tong Zhang, S. M. W. and Indurkhya, N.: 2002, Experiments in high-dimensional text categorization.
URL: *citeseer.ist.psu.edu/620104.html*

Fukumoto, F. and Suzuki, Y.: 2002, Manipulating large corpora for text classification, *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 196–203.

Fürnkranz, J.: 2003, Round robin ensembles., *Intell. Data Anal.* **7**(5), 385–403.

Galavotti, L., Sebastiani, F. and Simi, M.: 2000, Experiments on the use of feature selection and negative evidence in automated text categorization, *in* J. L. Borbinha and T. Baker (eds), *Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries*, number 1923, Springer Verlag, Heidelberg, DE, Lisbon, PT, pp. 59–68.
URL: *citeseer.ist.psu.edu/478428.html*

Garner, S.: 1995, Weka: The waikato environment for knowledge analysis.
URL: *citeseer.ist.psu.edu/garner95weka.html*

Ghani, R.: 2001, *Using error-correcting codes for efficient text classification with a large number of categories*, PhD thesis.

Giacinto, G. and Roli, F.: 1999, Methods for dynamic classifier selection, *Proceedings of the 10th International Conference on Image Analysis and Processing, Venice, Italy*, pp. 659–664.
URL: *citeseer.ist.psu.edu/giacinto99methods.html*

Granitzer, M.: 2003, *Hierarchical Text Classification using Methods from Machine Learning*, PhD thesis, Austria.

Grimaldi, M., Cunningham, P. and Kokaram, A.: 2002, Classifying music by genre using the wavelet packet transform and a round-robin ensemble, *Technical Report, TCD-CS-2002-64* .
URL: *http://www.cs.tcd.ie/publications/tech-reports/tr-index.02.html*

Grimaldi, M.. Cunningham, P. and Kokaram, A.: 2003, A wavelet packet representation of audio signals for music genre classification using different ensemble and feature selection techniques, *MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, ACM Press, New York, NY, USA, pp. 102–108.

Hayes, C., Cunningham, P. and Doyle, M.: 1998, Distributed cbr using xml, *Proceedings of the Workshop: Intelligent Systems and Electronic Commerce, Bremen*.
URL: *citeseer.ist.psu.edu/hayes98distributed.html*

Huang, Y.: 2003, Support vector machines for text categorization based on latent semantic indexing, *Technical report*, Electrical and Computer Engineering Department, The Johns Hopkins University.
URL: *citeseer.ist.psu.edu/huang01support.html*

Huang, Z., Chung, W., Ong, T.-H. and Chen, H.: 2002, A graph-based recommender system for digital library, *JCDL '02* pp. 65–73.

Huffman, D. A.: 1952, A method for the construction of minimum redundancy codes, *Proceedings of the Institute of Radio Engineers*, Vol. 40, pp. 1098–1101.

Iyer, R. D., Lewis, D. D., Schapire, R. E., Singer, Y. and Singhal, A.: 2000, Boosting for document routing, *in* A. Agah, J. Callan and E. Rundensteiner (eds), *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, ACM Press, New York, US, McLean, US, pp. 70–77.
URL: *citeseer.ist.psu.edu/iyer00boosting.html*

Joachims, T.: 1999, Text categorization with support vector machines: Learning with many relevant features, *Proceedings of the 10th ECML*, Springer.

Katirai, H.: 1999, Filtering junk E-mail: A performance comparison between genetic programming and naive bayes.
URL: *citeseer.ist.psu.edu/katirai99filtering.html*

Keller, M. and Bengio, S.: 2003, Textual data representation, *IDIAP-RR 74*, IDIAP.
URL: *ftp://ftp.idiap.ch/pub/reports/2003/rr03-74.pdf*

Keogh, E., Lonardi, S. and Ratanamahatana, C. A.: 2004, Towards parameter-free data mining, *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press, New York, NY, USA, pp. 206–215.

Kohavi, R. and John, G. H.: 1997, Wrappers for feature subset selection, *Artificial Intelligence* **97**(1-2), 273–324.
URL: *citeseer.ist.psu.edu/article/kohavi97wrappers.html*

Kolmogorov, A.: 1965, quantifing the randomness of strings.

Lam, S. L. and Lee, D. L.: 1999, Feature reduction for neural network based text categorization, *in* A. L. Chen and F. H. Lochovsky (eds), *Proceedings of DASFAA-99, 6th IEEE International Conference on Database Advanced Systems for Advanced Application*, IEEE Computer Society Press, Los Alamitos, US, Hsinchu, TW, pp. 195–202.
URL: *citeseer.ist.psu.edu/lam99feature.html*

Lang, K.: 1995, 20 newsgroup dataset.

Leake, D.: 1996, CBR in context: The present and future, *Technical report*.
URL: *citeseer.ist.psu.edu/article/leake96cbr.html*

Lenz, M., Hübner, A. and Kunze, M.: 1998, Question answering with textual CBR, *Lecture Notes in Computer Science* **1495**, 236+.
URL: *citeseer.ist.psu.edu/lenz98question.html*

Lewis, D. D.: 1992, Feature Selection and Feature Extraction for Text Categorization, *Proceedings of Speech and Natural Language Workshop*, Morgan Kaufmann, San Mateo, California, pp. 212–217.
URL: *citeseer.ist.psu.edu/lewis92feature.html*

Lewis, D. D.: 1998, Naive (Bayes) at forty: The independence assumption in information retrieval., *in* C. Nédellec and C. Rouveirol (eds), *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, Springer Verlag, Heidelberg, DE, Chemnitz, DE, pp. 4–15.
URL: *citeseer.ist.psu.edu/lewis98naive.html*

Lewis, D. D.: 1999, Reuters-21578 text categorization test collection distribution 1.0.
URL: *http://www.research.att.com/*

Lewis, D. D. and Ringuette, M.: 1994, A comparison of two learning algorithms for text categorization, *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US, pp. 81–93.
URL: *citeseer.ist.psu.edu/lewis94comparison.html*

Li, M., Chen, X., Li, X., Ma, B. and Vitanyi, P.: 2003, The similarity metric.
URL: *citeseer.ist.psu.edu/article/li03similarity.html*

Li, M. and Vitanyi, P. M. B.: 1993, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer-Verlag, Berlin.
URL: *citeseer.ist.psu.edu/li97introduction.html*

Li, Y. H. and Jain, A. K.: 1998, Classification of text documents, *THE COMPUTER JOURNAL* 41.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C.: 2002, Text classification using string kernels, *J. Mach. Learn. Res.* **2**, 419–444.

Loewenstern, D., Hirsh, H., Yianilos, P. and Noordewier, M.: 1995, Dna sequence classification using compression-based induction, *Technical Report 95-04, DIMACS* .

McNemar, Q.: 1947, Note on the sampling error of the difference between correlated proportions or percentages, *Psychometrika* **12**, 153–157.

Mitchell, T. M.: 1997, *Machine Learning*, McGraw-Hill, New York.

Mladenic, D.: 1998, Feature subset selection in text-learning, *Procedings of European Conference on Machine Learning*, pp. 95–100.
URL: *citeseer.ist.psu.edu/article/mladenic98feature.html*

Niblett, T.: 1987, Constructing decision trees in noisy domains., *Procedings of EWSL*, pp. 67–78.

Pal, M. and Mather, P. M.: 2001, Decision tree based classification of remotely sensed data, *Proceedings of ACRS 2001 - 22nd Asian Conference on Remote Sensing*, pp. 245–248.

Platt, J. C.: 1999, Fast training of support vector machines using sequential minimal optimization, *Advances in kernel methods: support vector learning* pp. 185–208.

Porter, M. F.: 1997, An algorithm for suffix stripping, *Readings in information retrieval* pp. 313–316.

Quinlan, J. R.: 1993, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Quinlan, J. R.: 1997, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Mateo, CA., USA.

Rennie, J.: 2001a, *Improving multi-class text classification with naive bayes*, PhD thesis.

Rennie, J.: 2001b, Improving multi-class text classification with support vector machine.
URL: *citeseer.ist.psu.edu/article/rennie01improving.html*

Rifkin, R. and Klautau, A.: 2004, In defense of one-vs-all classification, *J. Mach. Learn. Res.* **5**, 101–141.

Robinson, H.: 2003, Feature selection and representation in text.
URL: *citeseer.ist.psu.edu/707969.html*

Roth-Berghofer, T. R.: 2003, Knowledge maintenance of case-based reasoning systems – the SIAM methodology, *Zeitschrift KI - K"unstliche Intelligenz* **17**(1), 55–57.

Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E.: 1998, A bayesian approach to filtering junk E-mail, *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report WS-98-05, Madison, Wisconsin.
URL: *citeseer.ist.psu.edu/sahami98bayesian.html*

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D. and Stamatopoulos, P.: 2001, Stacking classifiers for anti-spam filtering of e-mail.
URL: *http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0106040*

Salton, G. and Buckley, C.: 1987, Term weighting approaches in automatic text retrieval, *Technical report*, Ithaca, NY, USA.

Salton, G. and Buckley, C.: 1988, Term-weighting approaches in automatic text retrieval., *Information Processing and Management* pp. 513–523.

Salton, G. and McGill, M. J.: 1986, *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., New York, NY, USA.

Schapire, R. E. and Singer, Y.: 2000, BoosTexter: A boosting-based system for text categorization, *Machine Learning* **39**(2/3), 135–168.
URL: *citeseer.ist.psu.edu/schapire00boostexter.html*

Schohn, G. and Cohn, D.: 2000, Less is more: Active learning with support vector machines, *Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 839–846.
URL: *citeseer.ist.psu.edu/schohn00less.html*

Scott, S. and Matwin, S.: 1999, Feature engineering for text classification, *in* I. Bratko and S. Dzeroski (eds), *Proceedings of ICML-99, 16th International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, US, Bled, SL, pp. 379–388.
URL: *citeseer.ist.psu.edu/scott99feature.html*

Sebastiani, F.: 2001, Organizing and using digital libraries by automated text categorization, *Proceedings of the AI*IA Workshop Artificial Intelligence for Digital Libraries and Cultural Heritage*, Bari, IT, pp. 93–94. Invited talk.
URL: *http://faure.iei.pi.cnr.it/ fabrizio/Publications/AIIA01.pdf*

Sebastiani, F.: 2002, Machine learning in automated text categorization, *ACM Comput. Surv.* **34**(1), 1–47.

Siroker, D. and Miller, S.: 2003, Topical clustering, summarization, and visualization.

Siyang, G., Qingrui, L., Lin, M., Seong, W. S. and Yongluan, Z.: 2000, Meta-classifier in text classification.
URL: *http://www.comp.nus.edu.sg/ zhouyong/courses.html*

Smyth, B. and Keane, M. T.: 1995, Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems, *Proceedings of IJCAI*, pp. 377–383.
URL: *citeseer.ist.psu.edu/smyth95remembering.html*

Smyth, B. and McKenna, E.: 1998, Modelling the competence of case-bases, *EWCBR '98: Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning*, Springer-Verlag, London, UK, pp. 208–220.

Tang, N.: 2001, Text categorisation using support vector machines.

Ting, K. M. and Witten, I. H.: 1999, Issues in stacked generalization, *Journal of Artificial Intelligence Research* **10**, 271–289.
URL: *citeseer.ist.psu.edu/ting99issues.html*

von Wangenheim, C.: 2000, Case-based reasoning - a short introduction, Springer.

Wang, M.-W., Nie, J.-Y. and Zeng, X.-Q.: 2005, A latent semantic classification model, *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, ACM Press, New York, NY, USA, pp. 261–262.

Watson, I.: 1998, *Applying case-based reasoning: techniques for enterprise systems*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Wess, S., Althoff, K.-D. and Derwand, G.: 1994, Using k-d trees to improve the retrieval step in case-based reasoning, *EWCBR '93: Selected papers from the First European Workshop on Topics in Case-Based Reasoning*, Springer-Verlag, London, UK, pp. 167–181.

Wiener, E. D., Pedersen, J. O. and Weigend, A. S.: 1995, A neural network approach to topic spotting, *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US, pp. 317–332.
URL: *citeseer.ist.psu.edu/wiener95neural.html*

Wilson, D. L.: 1972, Asymptotic properties of nearest neighbor rules using edited data, *IEEE Transactions on Systems, Man and Cybernetics 2(3)*, pp. 408–421.

Witten, I. H. and Frank, E.: 2005, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann Publishers Inc., San Francisco, CA., USA.

Yang, Y. and Pedersen, J. O.: 1997, A comparative study on feature selection in text categorization, *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 412–420.

Zhang, R. and Rudnicky, A. I.: 2003, Comparative study of boosting and non-boosting training for constructing ensembles of acoustic models, *Proceedings of EUROSPEECH 2003 - INTERSPEECH 2003 8th European Conference on Speech Communication and Technology*.

Zhou, Z.-H. and Yu, Y.: 2005, Adapt bagging to nearest neighbor classifiers., *J. Comput. Sci. Technol.* **20**(1), 48–54.

Ziv, J. and Lempel, A.: 1977, A universal algorithm for sequential data compression, *IEEE Transactions on Information Theory* **23**(3), 337–343.
URL: *citeseer.ist.psu.edu/ziv77universal.html*