

School of Computer Science and Software Engineering
Monash University

Bachelor of Computer Science with Honours (1608)
Literature Review Semester 2, 2002

**Text Classification with Labeled and Unlabeled
Data**

by

Aleksandar Milisic 12834556
milisic@mail.csse.monash.edu.au

Supervisor: Dr. David Albrecht
dwa@mail.csse.monash.edu.au

Contents

1	Abstract	1
2	Introduction	1
3	Classifying with Labeled Data	3
3.1	Naïve Bayes	3
3.2	K-Nearest Neighbour	4
3.3	PDDP and Naïve Bayes	4
3.4	Support Vector Machines	5
3.5	Multi-Class Support Vector Machines	7
3.5.1	One vs. One Classifier	7
3.5.2	One vs. Rest Classifier	8
3.5.3	Error-Correcting Output Codes	8
4	Classifying with Labeled and Unlabeled Data	9
4.1	Labeling the Unlabeled Data	9
4.1.1	Expectation-Maximization (EM) Algorithm	9
4.1.2	Co-training Algorithm	10
4.1.3	Transductive Support Vector Machines (TSVM)	11
4.2	Classifying and Clustering - A Combined Approach	12
4.2.1	Single Pass Algorithm and SVMs	12
4.2.2	SNOB	14
5	Feature Selection	15
5.1	Syntactic Indexing	15
5.2	Information Gain (IG)	15
5.3	Concept Indexing (CI)	16
5.4	Term Frequency - Inverse Document Frequency	16
6	Summary	18

1 Abstract

Recently, the amount of information available on the Internet and in other electronic forms has experienced a rapid growth. Unfortunately, humans are not very good at managing and utilizing huge collections of documents. As the amount of information stored in electronic form increases further, the need for developing tools to help people retrieve, filter and manage documents becomes more obvious.

Improving the performance of text classifiers, therefore, is of great importance in various segments of the industry (business, news) because the number of documents that are being handled on a daily basis is increasing rapidly by the day and becoming more and more daunting for humans to cope with.

For that reason text categorization has become an extremely popular area of research and a lot of work has been done investigating various approaches, their strengths, weaknesses and how to overcome them.

In the following sections I will give an overview of text classification, present and analyze the various methods used and proposed by researchers, as well as their disadvantages and potential room for improvement.

2 Introduction

With the growing number of documents in electronic form, text classification, the task of assigning documents to pre-defined categories, has inspired great interest in researchers around the world, who have developed various machine-learning algorithms for automating this task. These algorithms are commonly based on the *vector space model* (Kwok, 1999) where sparse vectors represent the text documents and each component corresponds to a feature extracted from the document. The features describe a document in a particular way, with one of the more common approaches having them indicate the presence of words in the document. This creates a *word presence vector* (see Figure 1). These algorithms are described in more detail in the following sections.

The main problems these classifiers encounter in text classification are:

- *High dimensional input spaces*: the document vectors are very high dimensional and in general have thousands of components representing the different features of a document collection. This often imposes huge computational and memory requirements on text classifiers, severely affecting their performance. This property, in turn, requires the use of *feature selection* (Lewis, 1992; Joachims, 1997) to reduce the dimensionality of the data set.

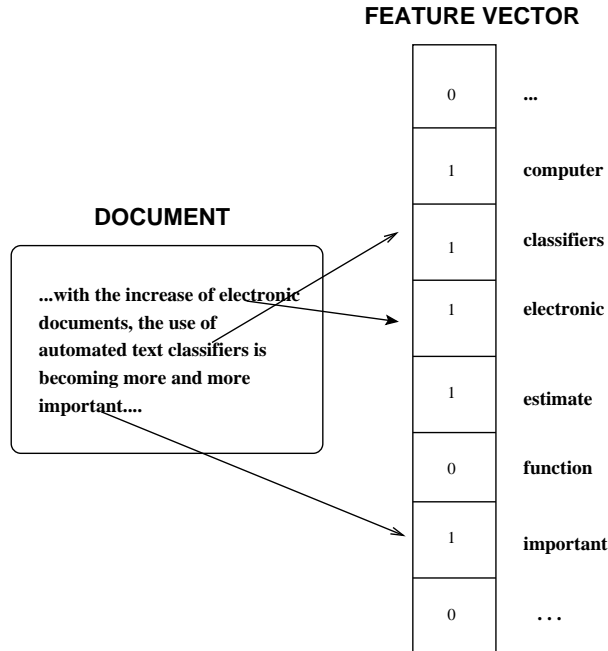


Figure 1: Representing Documents as Feature Vectors

- *Text classifiers must be able to learn from a small set of labeled examples:* labeling documents manually is an expensive and time-consuming task. This emphasizes the need for text classifiers to be able to learn from a small set of labeled and a large set of unlabeled documents without severely affecting accuracy. Often classifiers work with labeled data only, limiting their applicability to this domain. However, there also exist various techniques for successfully classifying data after learning from only a small set of labeled examples and a large set of unlabeled ones (Nigam, McCallum, Thrun and Mitchell, 2000; Joachims, 1999; Blum and Mitchell, 1998; Raskutti, Ferrá and Kowalczyk, 2002; Fang, Parthasarathy and Schwartz, 2001).
- *Multi-Class Classification:* this is a problem common in text classification. Text documents, for example news stories, can belong to more than one class (sports, IT, politics, health...). Various methods exist for dealing with this problem, and the particular area of my research will be the application of Support Vector Machines (SVM) to multi-class classification. Some solutions regarding SVMs already exist (Salomon, 2001; Lee, Lin and Wahba, 2001a), however they are not very efficient as I will describe later on.

Having these problems in mind, researchers have designed various algo-

rithms for tackling the problem of text classification. They can be divided into two major approaches:

- **Classifying with labeled data alone** and
- **Classifying with labeled and unlabeled data**

Another area of great importance in text classification is **feature selection**, the task of reducing the total number of features used in classification by discarding irrelevant features. Irrelevant features are those that, for example, appear in all classes approximately the same number of times, and therefore do not provide the classifier with any relevant information. Feature selection, therefore, helps reduce the problems associated with high dimensionality. Various classification and feature selection techniques are discussed in the following sections.

3 Classifying with Labeled Data

Text classification is a supervised learning task, which simply assigns labels to new documents using information obtained during learning from a set of labeled documents. The training set is labeled manually. A large number of approaches exist, ranging from probabilistic methods (Fang et al., 2001; Lewis and Ringuette, 1994), combinations of clustering methods and classifiers (Fang et al., 2001), k-Nearest Neighbour classifiers (Han, Karypis and Kumar, 2001) to Support Vector Machines(SVMs) (Joachims, 1998; Kwok, 1999; Dumais, 1998; Raskutti et al., 2002; Tong and Koller, 2000).

3.1 Naïve Bayes

The **Naïve Bayes** algorithm is widely used in text classification, and has proven to provide very good results (Fang et al., 2001; Lewis and Ringuette, 1994). For each document in the document collection, the algorithm computes the posterior probability that the document belongs to different classes. Then the document is assigned to the class with the highest probability.

When determining the class C of a document from a collection with unique words $w_1, w_2 \dots w_n$, Fang et al. (2001) use the expression

$$C = \operatorname{argmax} P(w_1, w_2 \dots w_n | C_j) P(C_j) \quad (1)$$

As Fang et al. (2001) explain, the problem with this approach is that the number of $P(w_1, w_2 \dots w_n | C_j)$ terms is equal to the number of possible classes multiplied by the number of possible documents. This, of course, would require an extremely large training set and is therefore not very efficient. For this reason, Naïve Bayes often makes the assumption of *word independence*.

In other words, it assumes that all words in a given class are probabilistically independent of each other. This simplifies the above equation to:

$$C = \operatorname{argmax} P(C_j) \prod_i P(w_i | C_j) \quad (2)$$

Even though in practice this assumption hardly ever holds, experiments show that the Naïve Bayes classifier still performs well (Fang et al., 2001; Lewis and Ringuette, 1994).

3.2 K-Nearest Neighbour

The K-Nearest Neighbour algorithm is based on finding the k nearest neighbours of a test document, where the “closeness” is expressed in terms of similarity. Then the similarities of the test document to the k neighbours are added up according to the class of the neighbours. After adding up these similarities, the test document is assigned to the most similar class, as measured by the k similarities added up. The commonly used similarity measure is the “cosine” similarity. The cosine similarity between documents X and Y from a collection with a set of unique words W is defined as:

$$\cos(X, Y) = \frac{\sum_{w \in W} X_w \cdot Y_w}{\sqrt{\sum_{w \in W} X_w^2} \cdot \sqrt{\sum_{w \in W} Y_w^2}} \quad (3)$$

One of the weaknesses of this approach is that it uses all features in computing the similarities, when in most cases only a subset of the features is useful for classification. This can lead to poor and not very accurate similarity measures, causing the performance to degrade.

For this reason, Han et al. (Han et al., 2001) propose a modified version of k-NN, **Weight Adjusted k-Nearest Neighbour (WAKNN)**. In this method each word from the training set is assigned an importance measure and a weight vector V is maintained storing the importance for all the words in the training set. The weight vector is used to compute the similarity between documents, where more important words (as shown in the weight vector), contribute more to the final result of the “similarity” computation. The weighted cosine similarity measure for documents X and Y is defined as:

$$\cos(X, Y, W) = \frac{\sum_{w \in W} (X_w \cdot V_w) \cdot (Y_w \cdot V_w)}{\sqrt{\sum_{w \in W} (X_w \cdot V_w)^2} \cdot \sqrt{\sum_{w \in W} (Y_w \cdot V_w)^2}} \quad (4)$$

However, this method also requires expensive computations when calculating the similarity between a new document and every training document.

3.3 PDDP and Naïve Bayes

A new approach to this problem is investigated by Fang et al. (2001), where a Naïve Bayes Classifier was used on information that was obtained by running

the Principal Direction Divisive Partitioning (PDDP) clustering algorithm on the training data. The clustering algorithm was used to find a small set of distinctive words that would be later used to train the Naïve Bayes classifier. The overall performance of the classifier showed an improvement of 85% compared to the Bayes classifier alone (Fang et al., 2001).

PDDP creates a binary tree, where each node has a set of documents, various figures calculated from these documents (for example, the cluster centroid) and pointers to two child nodes. The algorithm uses an $n \times m$ term frequency matrix, where n is the number of unique words in the document set and m is the number of documents. The term frequency matrix, together with the cluster centroid is used to split the documents in a node into two partitions. The PDDP tree root contains all the documents. Then each leaf cluster is recursively split into two children. This continues until some criterion is satisfied. In Fang et al.'s (2001) experiments, the splitting of clusters continued until “pure nodes” were obtained, which means nodes that contained documents from only one unique class. Which node to split at each stage is determined by using the “scatter values” of each cluster, where the scatter value is just a measure of how compact the cluster actually is.

The performance of the algorithm depends heavily on the size of the term frequency matrix. Document collections usually have thousands of unique words and the number of documents clustered are also often in the thousands. This makes the matrix extremely large and degrades performance. Fang et al. (2001) propose a solution which includes human experts finding the most important words and hence reducing the matrix size without severely affecting accuracy. However, this is a process that involves human involvement, and of course, that is something we would like to minimize in the area of text classification.

3.4 Support Vector Machines

Support Vector Machines(SVM) (Joachims, 1998; Kwok, 1999; Dumais, 1998; Raskutti et al., 2002; Tong and Koller, 2000; Schölkopf, 1998) are commonly used in text classification. Given a training set of labeled documents, they construct a hyperplane by trying to simultaneously maximize the margin between the sets of two classes (“positive” and “negative”) and minimize the classification error. The points that are found to belong to the planes supporting the two classes are called **support vectors** (see Figure 2).

The decision function is of the form,

$$f(x) = \text{sign}((w \cdot x) + b) \quad (5)$$

where w is a vector determining the orientation of the plane and b is the offset of the plane from the origin (Bennet and Campbell, 2000). The

function implies that a document is assigned to the positive class if $w \cdot x + b > 0$, otherwise it is assigned to the negative class.

The optimal solution for a training set is found by determining w and b (from Equation 5) such that:

$$\min \frac{1}{2} \|w\|^2 + \frac{C}{m} \sum_{i=1}^m \epsilon_i, \quad y_i(w \cdot x_i - b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0 \quad (6)$$

for all $i = 1, \dots, m$, where C is a constant. Since the margin between two classes is defined as $\frac{2}{\|w\|^2}$, by minimizing the first term in (6), the margin between the two classes will be maximized. The second term denotes the penalty for every training example that is misclassified (otherwise $\epsilon_i = 0$). When the data is linearly separable (i.e. all examples belong to the correct side of the hyperplane, as in Figure 2), then the penalty terms from equation (6) are omitted.

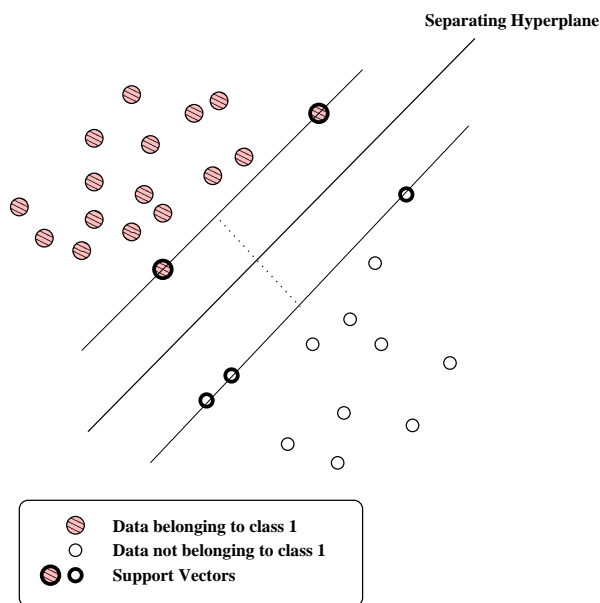


Figure 2: SVM solution for two linearly separable classes

Research so far indicates that SVMs are well suited for the text classification problem (Joachims, 2001; Bennet and Campbell, 2000). In his paper, Kwok (1999) explains that SVMs can handle dynamic document collections. They do that by using the decomposition algorithm (Kwok, 1999), which allows incorporating new documents into the existing set of support vectors (obtained from the training set) without losing any accuracy.

Also, Joachims (1998) indicates the property of SVMs that their learning process is independent of the dimensionality of the feature space (i.e. the

number of unique words). This means SVMs can generalize well even in the presence of many features if the data can be separated with a wide margin, making them less prone to *overfitting*.

Although, according to Joachims (1998), documents in general are linearly separable (with minor classification errors), even in the odd case of this not being applicable to a certain training set, SVMs can construct highly non-linear classification functions by using *kernels* (Bennet and Campbell, 2000). Kernels are functions used to replace the mappings from the original two-dimensional input space (for linear classifiers) to the multi-dimensional input space (for non-linear classifiers). They are used because the mapping functions are often impractical to compute while kernels have a much simpler form (Bennet and Campbell, 2000; Dumais, 1998). This property makes SVMs even more appropriate for use in text classification. For more information on Support Vector Machines and kernels, see (Schölkopf, 1998; Kwok, 1999; Bennet and Campbell, 2000; Dumais, 1998).

There are, however, certain issues regarding the use of SVMs in text classification:

- The algorithm for finding the support vectors in the learning process is quadratic in space. This obviously is a problem, since training is usually done with thousand of documents and possibly thousands of unique words, so the memory requirements can grow rapidly. Various feature selection techniques are being investigated to help reduce this problem. (For more information, see section 5).
- Document collections can have several various topics incorporated, and one document can belong to several topics (i.e. they are not mutually exclusive). This is a problem that in a certain way limits the use of SVMs in text classification. Several approaches exist to this problem, and they will be investigated in the following section.

For the above reasons, SVMs will be included in this project's research, particularly their application to dealing with multi-class classification problems and the benefits of combining them with other techniques.

3.5 Multi-Class Support Vector Machines

The **Multi-Class classification problem** for SVMs has been approached in different ways (Salomon, 2001; Lee et al., 2001a; Ghani, 2000; Schölkopf and Smola, 2002), each of them having their advantages and weaknesses.

3.5.1 One vs. One Classifier

The *One vs. One classifier* (Salomon, 2001) is a popular and simple technique. It creates a SVM for all possible combinations of classes. Then during

classification, a new document is run through all the SVMs and assigned to the class that “wins” the greatest number of these classifications. This is why this method is often called a “voting scheme”. An obvious advantage of this scheme is that even if a new example is misclassified by one of the SVMs, there is still a chance for it being classified correctly since there are $N - 1$ SVMs trained for a data set with N classes.

Weaknesses: A disadvantage is that if there are N classes, the number of trained SVMs is $N \times (N - 1) / 2$, which means the number of SVMs will grow rapidly with the number of different classes. This also makes classification slow, since the document has to be classified by all the SVMs before a final decision is made.

3.5.2 One vs. Rest Classifier

The *One vs. Rest classifier* (Salomon, 2001) builds N SVMs for a training set with N classes. Each SVM separates one class, C_i , from the rest of the classes in the training set. This is done for all classes from 1 to N . The basic idea is then to assign a point to the class whose hyperplane is furthest away. This method is less “expensive” than the One vs. One method for the simple reason that it needs less classifiers (N compared to $N \times (N - 1) / 2$).

Weaknesses: All the classes are involved in each of the trained SVMs, which makes the training process still a very time consuming one. Also, as Lee, Lin and Wahba (2001b), who give a probabilistic interpretation of SVMs, explain, it is often difficult to isolate one class from the rest, causing a decrease in classifier accuracy.

3.5.3 Error-Correcting Output Codes

Error-correcting output codes (ECOC) (Dietterich and Bakiri, 1995; Ghani, 2000; Schölkopf and Smola, 2002) are based on the use of codewords in data communications. They are used not only for multi-class SVMs, but in combination with other machine learning methods as well (Dietterich and Bakiri, 1995). The algorithm for an m -class problem can be explained as follows:

1. An $m \times n$ matrix M is created
2. Each class is assigned one n -length string (unique row) from the matrix
3. An SVM is trained for each column in the matrix (for a total of n SVMs)

where m is the number of unique classes and n is an arbitrarily chosen codeword length (having in mind that the rows in the matrix should be well separated to accommodate for errors, i.e. they should have some fixed

Hamming distance). In the testing phase each of the n SVMs are applied to the test example, each of them producing an output and contributing one digit to the n -length string. Finally, the test example is assigned to the class from M with the lowest Hamming distance (in other words, the class whose codeword is closest to the codeword produced by the n SVMs for that particular test example).

Weaknesses Although this method has produced good results, Schölkopf and Smola (2002) argue that it doesn't make enough use of the most important quantity in SVMs - the margin. Instead, as in the One vs. Rest method, it uses a variant of the "voting" scheme. Also, a major weakness not only in the ECOC method, but in all the multi-class approaches discussed here, is that they do not cater for the case when a single document belongs to multiple classes.

4 Classifying with Labeled and Unlabeled Data

The above-mentioned classifiers have the obvious disadvantage that they can't incorporate unlabeled documents in the learning process and utilize it for improving their accuracy. This also makes these techniques expensive since they require a large set of manually labeled documents. However, there do exist techniques for learning to classify documents from a small set of labeled and large set of unlabeled examples (Nigam et al., 2000; Joachims, 1999; Blum and Mitchell, 1998). This is of great importance since their capability of learning with a small set of labeled examples minimizes the need for manual labeling.

The majority of existing techniques tend to label the unlabeled data. However, recently a new technique has been investigated by Raskutti et al. (2002) which involves combining a clustering algorithm with a Support Vector Machine. Both approaches will be discussed in the following sections.

4.1 Labeling the Unlabeled Data

As already mentioned, the majority of techniques using unlabeled data in the training process tend to label the data. There exist numerous methods that have proven to be successful in this area and the following are summaries of the most popular and widely used approaches.

4.1.1 Expectation-Maximization (EM) Algorithm

A combination of the **Expectation-Maximization (EM) algorithm** and a Naïve Bayes classifier, as presented by Nigam et al. (2000), shows that the accuracy of classifiers can be improved by augmenting a small number of labeled training documents with a large collection of unlabeled documents.

The EM algorithm is widely used in many applications. It is used for finding the maximum likelihood in problems with incomplete data. In the case of text classification, the unlabeled documents are the incomplete data since they are missing the class labels. The EM algorithm proceeds as follows:

1. Estimate Naïve Bayes parameters, θ (includes the probability of a word given a class $\theta_{w_i|c_j}$ and the class prior probabilities θ_{c_j}), from the labeled documents
2. Loop until θ doesn't further change:
 - Use Naïve Bayes classifier to assign predicted class labels to each unlabeled document
 - Estimate new classifier parameters, θ , using the whole set of documents (initially labeled and labeled in the previous step)
3. Output a classifier, with parameters θ that can now label unlabeled documents by predicting the class label

Weaknesses: To achieve improvement with the EM algorithm, some assumptions about how the data are generated must be satisfied (for example, the Naïve Bayes classifier assumes words in text are independent of each other for a given class). However, in practice, text rarely complies with these assumptions, causing the algorithm's performance to actually deteriorate. In order to avoid that, extensions to the existing algorithm are suggested, with one of the possibilities being a weighting factor that influences the unlabeled data's contribution to parameter estimation in EM.

4.1.2 Co-training Algorithm

Another approach is described by Blum and Mitchell (1998), known as the **co-training algorithm**. They assume that there exist two redundant sets of labeled data and that both would be sufficient for successful classification if there was enough labeled data. In other words, there exist two sets of data (X_1, X_2) such that training two different classifiers $f_1(x)$ and $f_2(x)$ with either of the two data sets would produce the same results. This allows the training of two separate classifiers that after being initially trained use their predictions to label the much larger unlabeled data set. The predictions on unlabeled data, produced by both classifiers, are then used to increase the training set of the other classifier. This means that the redundancy of features allows learning two distinct classifiers that in turn can now train each other over unlabeled data. An experiment on classifying web pages, where a classifier was learned for finding home pages of various Computer Science departments in 4 American universities, showed that the use of unlabeled data actually provides superior performance compared to just using the small labeled set.

Weaknesses: As Blum and Mitchell point out, their model is an “over-simplification” of the real world, since in two data sets, it is likely to find parameters x_1 and x_2 such that $f_1(x_1) \neq f_2(x_2)$, i.e. that do not fit into the models assumption. Also, further research is needed since only few experiments have been done, and only on the data set mentioned above.

A variation of the co-training algorithm mentioned above is described in the work by Goldman and Zhou (2000). The main strength of this method is that there are no assumptions made about having two redundant views that are sufficient for successful classification as there are in Blum and Mitchell’s method. Instead, there are two learning algorithms that are both trained on the labeled data, and then each learner takes a subset of the unlabeled data and labels it for the other. This is repeated until there is no more data to be selected for labeling. In the end, the two resulting hypothesis are combined to give the final hypothesis. Experiments done by Goldman and Zhou have shown that this technique outperforms certain learning algorithms such as ID3 and HOODG.

Weaknesses: At one stage, the algorithm has to determine which samples should algorithm A label for algorithm B and vice versa. The basic rule is that A should label a document for B only if A’s confidence in its label for the document is greater than B’s. The problem is that their experiments show that too high of a confidence-interval doesn’t allow enough data to be labeled, while confidence intervals that are too low allow too much data to be labeled. Experiments have been done using a 95% confidence interval, however there is no real proof as yet that this is the optimal interval.

4.1.3 Transductive Support Vector Machines (TSVM)

The method introduced by Joachims (Joachims, 1999) involves the use of **Transductive Support Vector Machines (TSVM)**. TSVMs are a flavour of SVMs with the difference that instead of searching for a decision function with a low error rate on the whole distribution of the training set, it attempts to classify a “test set” with as few errors as possible. The main idea is that the TSVM begins with a labeling of the test data based on the classification of an SVM, and then it improves the solution by switching the labels of the test examples, which is continuously done until a hyperplane that separates both the training and test data with maximum margin is found. The major advantage of this technique is that it doesn’t require a lot of labeled examples. Experiments performed by Joachims on the Reuters-21578 document collection, and verified using the precision-recall method¹(Joachims, 1999), show an improvement in the average of the

¹The Precision/Recall-Breakeven point is commonly used for assessing the performance of text classifiers. Precision is the probability that a document predicted to be in class ‘A’

precision-recall breakeven points from 48.4 for SVMs to 60.8 for TSVMs. The classifying was done using only 17 labeled examples and 3299 unlabeled ones. It should be noted, however, that the same unlabeled set used for training the TSVM was also used in the testing phase, an approach not very common in verifying the performance of text classification techniques.

Weaknesses: TSVMs still suffer from the same memory requirement problems as SVMs which further emphasizes the need for feature selection. They also have the problem of dealing with multi-class classification. In fact, TSVMs have to deal with the additional burden that as the number of unique classes increases, the number of different ways the test samples can be labeled increases exponentially. This, obviously, has serious performance implications.

This technique is still, however, very appealing simply because of its ability to learn with a very small number of labeled examples. For this reason, it will be further investigated in my research, particularly its application to classifying when combined with clustering methods, as is described in the next section.

4.2 Classifying and Clustering - A Combined Approach

Clustering is a method that assigns items (in this case documents) to automatically created groups based on a calculation of the degree of similarity between items and groups (clusters). It is a popular technique because it can group similar documents into clusters even without any labeled examples (classifying it as an “unsupervised learning” technique). Its role is to cluster the documents into similar groups and discover additional information about the properties (for example, similarity) of the documents in the document set. This information is then utilized by a classifier, which in most cases improves performance of the classifier (Raskutti et al., 2002; Fang et al., 2001). Several methods implementing this approach exist. A particularly interesting method, described by Raskutti et al. (2002), will be one of the major areas of investigation included in this project.

The next two sections describe the approach adopted by Raskutti et al. (2002) as well as Snob (Wallace and Dowe, 1994; Patrick, 1991; Oliver, Baxter and Wallace, 1996), a clustering method that could potentially improve text classification when combined with classifiers.

truly belongs to the class. Recall is the probability that a document belonging to class 'A' is classified into this class. The Precision/Recall-Breakeven point is the value for which precision and recall are equal.

4.2.1 Single Pass Algorithm and SVMs

The clustering algorithm used by Raskutti et al. (2002) is a modified version of the one described by Rasmussen (Rasmussen, 1992), a simple and popular clustering technique known as the *Single Pass Method*. The clustering algorithm used by Raskutti et al. (2002) can be described in the following way:

1. The first cluster C_1 is initialized with the first document D_1
2. For document D_i , the distances between document D_i and all the existing clusters are calculated and the minimum distance $Dist_{min}$ is found
3. If $Dist_{min}$ is less than some threshold T , the document is added to the corresponding cluster, otherwise a new cluster is initialized with document D_i
4. Loop until every document is allocated to a cluster

It should be noted that only the training data (including both labeled and unlabeled examples) is clustered. After clustering, new features are calculated and added to the labeled training documents as well as to the test documents. These new features include the document's cosine similarity (Equation 3) to the cluster's labeled centroid, unlabelled centroid, negative centroid, positive centroid etc. (Raskutti et al., 2002). When calculating these similarities, only the N most populated clusters are taken into account, where N is pre-determined by the user. After having new features added, the labeled training documents are used to train the classifier (in this case a SVM), which in turn provides better results since the SVM now has more information that describes the training set.

According to Raskutti et al. (2002), this method also has the advantage of being able to handle frequent additions to the document collection without re-training. Experiments using this clustering technique have been performed on SVMs and they show an improvement in the performance of SVMs even if the labeled set is only 0.25% of the total training set. It must be noted, however, that experiments were done using the word presence matrix (another feature representation technique might improve results).

Although they have advantages, these combined methods also suffer from the disadvantages of both the clustering algorithm used and the classifier. In this approach, the method encounters the same problems as SVMs, as described earlier, as well as the problems with the Single Pass Method.

The Single Pass Method, even though popular due to its simplicity, raises a few questions:

- First, the result of the clustering depends on the order in which the documents are processed (i.e. starting from somewhere in the middle of the document set would produce very different results).

- The choice of the threshold T can produce different results, so there is the issue of determining the optimal threshold for a particular data set. For a T that is too small, we end up with a lot of small clusters, while a T that is too large might produce only one large cluster which does not provide any information. According to Raskutti et al. (2002), the optimal threshold can be found by repeatedly clustering the documents but this comes at a performance cost.
- The number of clusters that are used to add features to the training and test set is pre-determined, but there is no guarantee that the chosen number is a true representative of the document population. A smaller or greater number of clusters might improve results, however this method leaves it to the user to test various possibilities.

As it is now, Raskutti et al.’s (2002) method adds cluster parameters to labeled data in the training set, which is later used to train an SVM. However, interesting results could be obtained by combining this clustering algorithm with a TSVM. This would mean adding cluster parameters to the unlabeled data from the training set as well and using it for training the TSVM. This will be one of the main areas of investigation in this project’s research.

4.2.2 SNOB

Snob (Wallace and Dowe, 1994; Patrick, 1991; Oliver et al., 1996) is a clustering program based on the Minimum Message Length (MML) (Wallace and Dowe, 1999; Wallace and Dowe, 1997) criterion that attempts to create the “best” classification (clustering) possible for a given collection of data (objects) with their attributes. MML attempts to minimize the length of a message $Message_{length}(H, D)$ consisting of two parts:

$$Message_{length}(H, D) = Message_{length}(H) + Message_{length}(D|H) \quad (7)$$

where H denotes a hypothesis that explains the data D . MML follows “Ockham’s razor” which says that “if two theories explain the facts equally well then the simpler theory is to be preferred” (William of Ockham)². In the same sense MML tries to find a representation of the given data that minimizes the length of the message needed to represent it. For more information on MML see (Wallace and Dowe, 1999; Wallace and Dowe, 1997).

Even though there doesn’t seem to be any research on the application of Snob to document classification, it is an approach that I will be investigating in my research. The reasons are the impressive results Snob has shown in other classification problems (Dowe, Allison, Dix, Hunter, Wallace and

²<http://www.csse.monash.edu.au/~lloyd/tildeMML/>

Edgoose, 1996; Kissane, Bloch, Burns, Patrick, Wallace and McKenzie, 1994; Oliver et al., 1996), as well as some advantages over the more widely used clustering techniques (Patrick, 1991). According to Patrick (1991), these advantages are:

1. Most clustering algorithms use some measure of similarity in order to create groups of similar objects. However, there is no theoretical proof favouring one similarity measure over others. Snob, on the other hand, attempts to group the objects so that the length of the message describing them is minimized.
2. As already mentioned for the Single Pass Algorithm, most clustering techniques fail in determining the optimal number of clusters to describe the given data set.

A weakness with this approach, at least when it comes to applying it in text classification, is that Snob hasn't been used in classifying objects with thousands of features (which is the case with documents). This could seriously limit the applicability of Snob to document classification. Dealing with this problem and determining therefore Snob's potential for use in text classification is going to be one of the issues investigated in this project's research.

For more detailed information on Snob see (Patrick, 1991; Wallace and Dowe, 1994; Wallace and Dowe, 1996).

5 Feature Selection

As becomes obvious after analyzing the various clustering and classifying techniques used in text categorization, the fact that document sets usually have thousands of unique words (features) often severely affects performance. Dealing with thousands of features requires expensive computations as well as large amounts of memory storage. Also, as Joachims (1997) explains, often a subset of features can help discriminate between classes better, while including redundant features in the classification can add noise and reduce performance. For these reasons, researchers are investigating various ways of representing features that will allow discarding a significant number of irrelevant features without affecting the accuracy of text classifiers.

5.1 Syntactic Indexing

One of the more recent approaches is presented by Lewis (1992), who tries to investigate the property of text that words often are not probabilistically independent. That is, some words combined provide valuable information in the context of the document topic (for example, "Mechanical Engineering"),

but singled out they don't provide almost any information at all. For that reason, he describes features as combinations of words in particular syntactic relationships and calls such features *syntactic indexing phrases*. He considers finding these by using clustering techniques to detect related features. Groups of these features are then replaced by a single feature which represents a sum of the weights of the corresponding words grouped. He refers to this strategy as *term clustering*. However, experiments performed using this technique showed that the improvement in performance was insignificant. This was caused by the fact that the document collection wasn't large enough to give a significant measure of the distributional properties of phrases. Also, the clustering techniques provided poor quality clusters, failing to capture a significant number of relevant and high quality semantic relationships. Another unexpected problem is that it appears that phrases, even though providing more information than single words, didn't turn out to be good content indicators. However, research using this method is still being investigated and new approaches are being considered.

5.2 Information Gain (IG)

Information Gain is a frequently used feature selection technique. Yang and Pedersen (1997) describe it's main strength as the fact that it measures the amount of information given by a word over all classes by considering both the presence and absence of the word from each class. In other words, it doesn't just use the presence of words in evaluating their importance, but also their absence is used to obtain information. The information gain of a term (word) t is defined as (Yang and Pedersen, 1997):

$$G(t) = - \sum_{i=1}^m P(c_i) \log P(c_i) + P(t) \sum_{i=1}^m P(c_i|t) \log P(c_i|t) + P(\bar{t}) \sum_{i=1}^m P(c_i|\bar{t}) \log P(c_i|\bar{t}) \quad (8)$$

where \bar{t} denotes the absence of the term and c_1, \dots, c_m denotes the m classes. In their experiments, Yang and Pedersen (1997) have discovered that the accuracy of classifiers is still very high (and in some cases improves) when using this selection technique. In some cases, however, the computational costs of this technique can be very high. Yang and Pedersen (1997) claim that estimating the probabilities has a time complexity of $O(N)$ and a space complexity of $O(VN)$ where N is the number of documents used for training and V is the number of unique words in the document collection. This, of course, causes the algorithm to become very inefficient for large document collection and vocabulary sizes.

5.3 Concept Indexing (CI)

Han and Karypis (2000) introduce a new feature selection technique called **Concept Indexing (CI)**. The main idea behind CI is to compute k -dimensional representations of all the documents, where k is a number much smaller than the initial dimensionality of the training set. This is achieved by doing the following for a document collection with n documents and m unique words:

- The documents are clustered into k groups (clusters)
- An $m \times k$ matrix C is created consisting of the k cluster centroids
- The k -dimensional representation of each document is obtained by calculating the expression $d.C$

Initial experiments show that this feature selection method achieves results comparable to other techniques and in the case of the k -NN classifier also greatly improves performance (Han and Karypis, 2000). It must be noted that this technique, when clustering, uses all the features for its calculations which in the case of large vocabulary sizes can slow down the execution time considerably. A possible remedy could be applying the TF-IDF model (as described in the following section) before the clustering in order to reduce the number of features.

5.4 Term Frequency - Inverse Document Frequency

Joachims (1997) describes an approach that describes features taking into account both the frequency in a single document and in the whole document collection. This is the **Term Frequency - Inverse Document Frequency (TF-IDF)** approach. It builds upon one of the more basic approaches, the *Term Frequency* method, by also considering the frequency of the word in the document set. The inverse document frequency is low if the word appears in many documents and highest if it appears in only one (see Equation 9). This accommodates the idea that less frequent words provide more information and are better in discriminating between classes. For these reasons, as well as for its simplicity, it is a very popular and widely used feature selection method. The TF-IDF value of a word w in a document d is defined as:

$$TF - IDF(w) = TF(w) \times \log \frac{|D|}{DF(W)} \quad (9)$$

where $TF(w)$ is the number of times the word w appears in document d , $DF(w)$ is the total number of documents it appears in (the **document frequency**) and $|D|$ is the size of the document collection.

Joachims (1997) in his experiments discovered that this method greatly improved the performance of the Rocchio algorithm (Joachims, 1997). ‘ For the above mentioned reasons, this approach will be the target of my research. The main issue to tackle is the fact that this representation fails to consider the importance of the position of a word within a document. Namely, a word appearing in a document title, heading or section title certainly must provide more information about the topic than a word somewhere in the middle of the text. The remedy for this could be giving words different weights according to their position in the document. This, however, could be difficult since the documents from the Reuters-21578 collection (Joachims, 1998), which was planned to be the collection used for this project, have only mostly brief “titles” and no other indicators of how important a word might be. This could seriously limit this proposed modification of the TF-IDF model.

6 Summary

The significance of text classification is well illustrated by the amount of work that has been done in this area, as well as the fact that it is still being investigated by researchers around the world. The techniques described in this paper are either the most recent ones or ones that are particularly interesting because of their impressive results in text classification.

Some of the techniques draw particular interest since they are very new and promising approaches. Also, different questions and possibilities arise from these techniques, which will be the area of investigation in this project's research. These issues can be summarized as follows:

1. Investigate whether Raskutti et al.'s (2002) approach to text classification (combining the Single Pass Algorithm with SVMs) can be as successful when applied to TSVMs.
2. Explore the potential of applying Snob to text classification and possibly combining it with SVMs and TSVMs.
3. Even though widely used, the existing SVM techniques for handling multi-class classification problems (Salomon, 2001; Lee et al., 2001a; Schölkopf and Smola, 2002) have several weaknesses. Possible extensions to the SVM methodology that could improve on these weaknesses should be investigated.
4. Try improving on TF-IDF as a feature selection technique by giving different weights to words according to their position in the document. This variant could hopefully allow reducing the number of features without degrading classifier performance, even though the potential weaknesses of this approach as explained in Section 5.4 should be noted.

References

- Bennet, K. P. and Campbell, C. (2000). Support vector machines: Hype or hallelujah?, *SIGKDD Explorations* **2**(2): 1–13.
- Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training, *COLT: Proceedings of the Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers.
- Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research* **2**: 263–286.
- Dowe, D. L., Allison, L., Dix, T. I., Hunter, L., Wallace, C. S. and Edgoose, T. (1996). Circular clustering of protein dihedral angles by Minimum Message Length., *Pacific Symposium on Biocomputing '96*, World Scientific, pp. 242–255.
- Dumais, S. (1998). Using svms for text categorization, *IEEE Intelligent Systems Magazine, Trends and Controversies, Marti Hearst*, pp. 21–23.
- Fang, Y. C., Parthasarathy, S. and Schwartz, F. (2001). Using clustering to boost text classification, *To appear in the ICDM Workshop on Text Mining (TextDM'01)*.
- Ghani, R. (2000). Using error-correcting codes for text classification, in P. Langley (ed.), *Proceedings of ICML-00, 17th International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, US, Stanford, US, pp. 303–310.
- Goldman, S. and Zhou, Y. (2000). Enhancing supervised learning with unlabeled data, *Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 327–334.
- Han, E.-H., Karypis, G. and Kumar, V. (2001). Text categorization using weight adjusted k-nearest neighbor classification, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 53–65.
*citeseer.nj.nec.com/han99text.html
- Han, E.-H. S. and Karypis, G. (2000). Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization, *Technical report*, University of Minnesota, Department of Computer Science/Army HPC Research Center, Minneapolis, USA.
- Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, *In Proceedings of the Fourteenth International Conference on Machine Learning*.

- Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features, in C. Nédellec and C. Rouveirol (eds), *Proceedings of ECML-98, 10th European Conference on Machine Learning*, Springer Verlag, Heidelberg, DE, Chemnitz, DE, pp. 137–142.
*citeseer.nj.nec.com/joachims98text.html
- Joachims, T. (1999). Transductive inference for text classification using support vector machines, *Proc. 16th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 200–209.
- Joachims, T. (2001). A statistical learning model of text classification with support vector machines, in W. B. Croft, D. J. Harper, D. H. Kraft and J. Zobel (eds), *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, ACM Press, New York, US, New Orleans, US, pp. 128–136.
- Kissane, D. W., Bloch, S., Burns, W. I., Patrick, J. D., Wallace, C. S. and McKenzie, D. P. (1994). Perceptions of family functioning and cancer, *Psycho-oncology*, Vol. 3, pp. 259–269.
- Kwok, J. T.-Y. (1999). Automated text categorization using support vector machine, *Proceedings of the International Conference on Neural Information Processing (ICONIP)*, Kitakyushu, Japan, pp. 347–351.
*citeseer.nj.nec.com/kwok98automated.html
- Lee, Y., Lin, Y. and Wahba, G. (2001a). Multicategory support vector machines, *Prepared for the Proceedings of the Conference, the 33rd Symposium on the Interface held in Costa Mesa, Ca.*
- Lee, Y., Lin, Y. and Wahba, G. (2001b). Multicategory support vector machines, *Technical report*, University of Wisconsin, 1210 West Dayton St., Madison, WI 53706.
- Lewis, D. D. (1992). Feature selection and feature extraction for text categorization, *Appeared in Speech and Natural Language: Proceedings of a workshop held at Harriman, New York*, Morgan Kaufmann, San Mateo, pp. 212–217.
- 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.
- Nigam, K., McCallum, A. K., Thrun, S. and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM, *Machine Learning* **39**(2/3): 103–134.

- Oliver, J. J., Baxter, R. A. and Wallace, C. S. (1996). Unsupervised learning using MML, *Machine Learning: Proceedings of the Thirteenth International Conference (ICML 96)*, Morgan Kaufmann Publishers, pp. 364–372.
- Patrick, J. D. (1991). Snob: A program for discriminating between classes, *Technical Report 91/151*, Department of Computer Science, Monash University, Clayton, Victoria, Australia.
- Raskutti, B., Ferrá, H. and Kowalczyk, A. (2002). Using unlabelled data for text classification through addition of cluster parameters, *In International Conference on Machine Learning (Accepted)*.
- Rasmussen, E. (1992). Clustering algorithms, *In William B. Frakes and Ricardo Baeza-Yates, editors, Information Retrieval: Data Structures and Algorithms*, Prentice Hall, Englewood Clis, New Jersey, pp. 419–442.
- Salomon, J. (2001). Support vector machines for phoneme classification, *Master of Science, School of Artificial Intelligence, Division of Informatics, University of Edinburgh*.
- Schölkopf, B. (1998). SVM's - a practical consequence of learning theory, *IEEE Intelligent Systems Magazine, Trends and Controversies, Marti Hearst*, pp. 18–21.
- Schölkopf, B. and Smola, A. (2002). *Learning with Kernels*, MIT Press, Cambridge, MA.
- Tong, S. and Koller, D. (2000). Support vector machine active learning with applications to text Classification, *Technical report*, Computer Science Department, Stanford University, Stanford, CA 94305, USA.
- Wallace, C. S. and Dowe, D. L. (1994). Intrinsic classification by MML - the Snob program, *Proc. 7th Australian Joint Conference on Artificial Intelligence, World Scientific*, pp. 37–44.
- Wallace, C. S. and Dowe, D. L. (1996). *Snob: Program for classification, mixture modelling, clustering, taxonomy, unsupervised pattern recognition*.
[*ftp://ftp.cs.monash.edu.au/software/snob/snob.documentation](ftp://ftp.cs.monash.edu.au/software/snob/snob.documentation)
- Wallace, C. S. and Dowe, D. L. (1997). MML mixture modelling of multi-state, Poisson, von Mises circular and Gaussian distributions, *Proc. 6th International Workshop on Artificial Intelligence and Statistics*, pp. 529–536. A short (1996) abstract is in Proc. Sydney International Statistical Congress (SISC-96), p197; and also in IMS Bulletin (1996), 25 (4), p410.

- Wallace, C. S. and Dowe, D. L. (1999). Minimum Message Length and Kolmogorov complexity, *The Computer Journal* **42**(4): 270–283.
- Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization, in D. H. Fisher (ed.), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, US, Nashville, US, pp. 412–420.