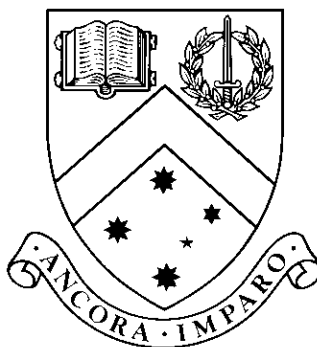


Text Classification with Labelled and Unlabelled Data

by

Aleksandar Milisic, BCompSc



Thesis

Submitted by Aleksandar Milisic

in partial fulfillment of the Requirements for the Degree of

Bachelor of Computer Science with Honours (1608)

in the School of Computer Science and Software Engineering at

Monash University

Monash University

November, 2002

© Copyright

by

Aleksandar Milisic

2002

Contents

List of Figures	v
Abstract	vii
Acknowledgments	ix
1 Introduction	1
2 Support Vector Machines (SVMs)	4
2.1 Multi-Class Support Vector Machines	6
2.1.1 One vs. One Classifier	6
2.1.2 One vs. Rest Classifier	6
2.2 Transductive Support Vector Machines	7
2.2.1 How does Unlabelled Data Improve Classification?	7
2.2.2 The TSVM Approach	8
3 Clustering - Utilizing the Unlabelled Data	10
3.1 Single Pass Algorithm	11
3.1.1 Implementation	12
3.2 Snob	13
3.2.1 Snob in Text Classification	14
4 Adding Features Relative to Clusters	17
4.1 Features Describing the Training Set	17
4.2 A New Set of Features	18

4.3	SVMs and TSVMs with Cluster Features - Motivation and Methods	20
5	Experiment Design	23
5.1	Aim	23
5.2	Data Sets	24
5.3	Evaluation	25
5.4	Partitioning the Data	26
6	Results	28
6.1	SVM vs SVM+Cluster	28
6.2	Discussion	30
7	Conclusions	39
8	Future Work	41
	Accuracy Scores	45
	Feature Distrubitions in TF-IDF Reuters	52
	References	54

List of Figures

1.1	Representing Documents as Feature Vectors	2
2.1	SVM solution for two linearly separable classes	5
2.2	Set of documents showing typical text structure	7
2.3	Example of TSVM	8
4.1	The format of an augmented feature vector	18
4.2	Structure of added cluster features	20
4.3	SVM using augmented feature vectors as training data	22
6.1	SVM probabilistic scores for 1% labelled data on TF-IDF Reuters	30
6.2	SVM probabilistic scores for 5% labelled data on TF-IDF Reuters	31
6.3	SVM probabilistic scores for 10% labelled data on TF-IDF Reuters	32
6.4	TSVM probabilistic scores for 1% labelled data on TF-IDF Reuters	33
6.5	TSVM probabilistic scores for 5% labelled data on TF-IDF Reuters	34
6.6	TSVM probabilistic scores for 10% labelled data on TF-IDF Reuters	35
6.7	SVM probabilistic scores for 1% labelled data on modApte Reuters split	35
6.8	SVM probabilistic scores for 5% labelled data on modApte Reuters split	36
6.9	SVM probabilistic scores for 10% labelled data on modApte Reuters split	36
6.10	SVM probabilistic scores for 1% labelled data on TF-IDF Reuters with new cluster features	37
6.11	SVM probabilistic scores for 5% labelled data on TF-IDF Reuters with new cluster features	37
6.12	SVM probabilistic scores for 10% labelled data on TF-IDF Reuters with new cluster features	38

8.1	Example of a multi-labelled document	43
2	SVM accuracy scores for 1% labelled data on TF-IDF Reuters	45
3	SVM accuracy scores for 5% labelled data on TF-IDF Reuters	46
4	SVM accuracy scores for 10% labelled data on TF-IDF Reuters	46
5	SVM accuracy scores for 1% labelled data on modApte Reuters split	47
6	SVM accuracy scores for 5% labelled data on modApte Reuters split	47
7	SVM accuracy scores for 10% labelled data on modApte Reuters split . . .	48
8	TSVM accuracy scores for 1% labelled data on TF-IDF Reuters	48
9	TSVM accuracy scores for 5% labelled data on TF-IDF Reuters	49
10	TSVM accuracy scores for 10% labelled data on TF-IDF Reuters	49
11	SVM accuracy scores for 1% labelled data on TF-IDF Reuters with new cluster features	50
12	SVM accuracy scores for 5% labelled data on TF-IDF Reuters with new cluster features	50
13	SVM accuracy scores for 10% labelled data on TF-IDF Reuters with new cluster features	51
14	Feature distribution of feature no. 13 from TF-IDF Reuters	52
15	Normal probability plot for non-zero values of feature no. 13 before trans- formation	53
16	Normal probability plot for non-zero values of feature no. 13 after transfor- mation	53

Text Classification with Labelled and Unlabelled Data

Aleksandar Milisic, BCompSc(Hons)
Monash University, 2002

Supervisor: Dr. David Albrecht

Abstract

The need for automated text classifiers is becoming more obvious with the rapid growth of information stored in electronic form. Handling and managing this information in an efficient manner is of vital importance to many sectors of the industry. For this reason, various text classification techniques have become areas of extensive research in recent years. This thesis explores and discusses the possibilities of utilizing unlabelled documents in text classification, since these documents are easier and cheaper to obtain than labelled ones. A recent approach of clustering labelled and unlabelled documents and adding features based on the created clusters is investigated. Support Vector Machines are used to classify the augmented document set. The thesis investigates this approach using various clustering algorithms and data sets. Also, an extension to Support Vector Machines that enables documents to be assigned to multiple categories (multi-labelled classification) is proposed.

Results show that Support Vector Machines are very sensitive to added cluster features so extreme care must be taken when augmenting the training set. The best results are obtained when the additional features are derived from well separated clusters. Also, in the thesis it is discovered that Support Vector Machines perform well when the cluster features are spread among different categories in a similar way the original features are.

Text Classification with Labelled and Unlabelled Data

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Aleksandar Milisic
November 10, 2002

Acknowledgments

I would like to thank my supervisor, Dr. David Albrecht, for all his support, patience and advice throughout the year.

Aleksandar Milisic

Monash University
November 2002

Chapter 1

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 among researchers around the world. This has influenced the development of various machine-learning algorithms for automating the task of text classification. 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* (Figure 1.1). In training sets, these feature vectors also have a label. This label indicates whether the document belongs to the class for which the classifier is being trained or not (this is often done by labeling the document with a “1” if it belongs to the class or “-1” otherwise).

For better understanding of the problem, an example application will be considered. A “news filter” will be our case example, an application that incorporates a text classifier trained to recognize articles of interest to us. If we are subscribed to several sources of articles and receive them via e-mail, but later decide that we only want to read articles on certain topics, we can train our classifier to accept only those articles of interest. For example, we might want to read only articles related to “IT” and ignore others. When given a new document, the application classifies it as either “IT” or “not IT”, and depending on the result decides whether to keep it or not.

The classifier must first be trained with a set of example articles (represented as feature vectors) of which one group belongs to the class “IT” and the other one doesn’t. The text classifier then takes the feature vectors and processes them depending on the classifier’s algorithm. After the training process, the application is ready to run and test new examples.

One limitation of using such automated text classifiers is the fact that they usually require a large amount of labelled documents in order to learn accurately. A large number of such approaches exist, ranging from probabilistic methods (Fang, Parthasarathy and

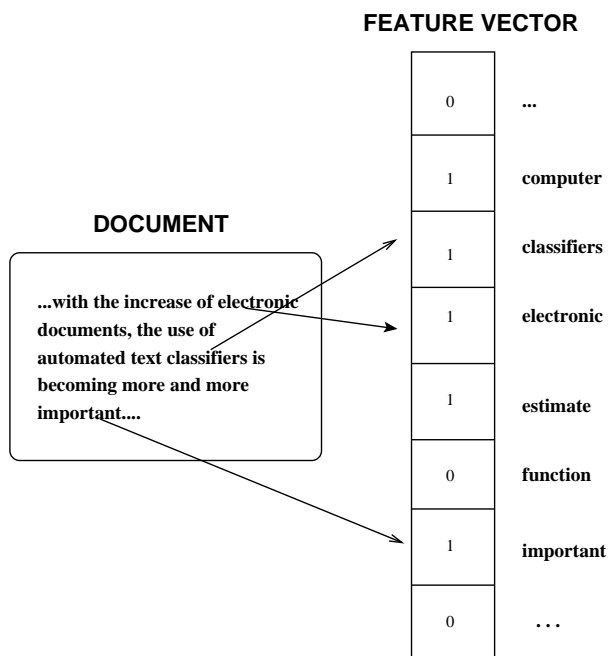


Figure 1.1: Representing Documents as Feature Vectors

Schwartz, 2001; Lewis and Ringuette, 1994), k-Nearest Neighbour classifiers (Han, Karypis and Kumar, 2001) to Support Vector Machines (SVMs) (Joachims, 1998; Kwok, 1999; Dumais, 1998; Raskutti, Ferrá and Kowalczyk, 2002; Tong and Koller, 2000). The problem arises from the fact that labeling documents is done manually and is a time-consuming and expensive task, especially when the number of documents is exceptionally large. For this reason recent research has been aimed at developing methods for training classifiers with a small set of labelled documents and a large set of unlabelled documents. In other words, we want our “news filter” to be able to classify accurately after learning from an enlarged set of examples of which a significant number are not labelled. Recent research has introduced many promising techniques for achieving this (Nigam, McCallum, Thrun and Mitchell, 2000; Joachims, 1999b; Blum and Mitchell, 1998; Raskutti et al., 2002). The research of this project draws special attention to the use of Support Vector Machines (SVMs), Transductive Support Vector Machines (Joachims, 1999b) and their combinations with clustering methods which are used to utilize the given unlabelled data (Raskutti et al., 2002). The clustering algorithms are used to assign subsets of the training set to similar groups and then add features to the training and test documents based on their relationships with the created groups.

Another limitation encountered with the use of text classifiers is that they often act as “binary classifiers” in the sense that their output only indicates whether a document belongs

to a certain class or not. However, documents (especially news articles, web pages etc.) can often belong to several categories. For example, news articles can belong to both categories “sports” and “politics”. For this reason, it is often desirable for the text classifier to indicate that the document could belong to several classes instead of “forcing” it to be labelled as one of them. In our given example, we would maybe want to check if the news article received belongs to other categories apart from “IT”. For example, the article could also contain information about the economy (therefore, belonging both to “business” and “IT” categories). Our news filter will label the document as one or the other, and if it decides the article belongs to the “business” category it will be lost. However, we would want the classifier to be able to detect the “multi-labelled” nature of the article and hence accept it so we can read it. Various methods that allow SVMs to handle data sets incorporating multiple classes already exist (Salomon, 2001; Lee, Lin and Wahba, 2001a; Ghani, 2000; Schölkopf and Smola, 2002), however their output is still in the above described format, therefore not recognizing the possibility of multiple labels being assigned to a single document. In this thesis an extension to SVMs is proposed that would handle this very common case in text classification (Chapter 8).

The thesis is organized as follows:

- **Chapter 2** gives an overview of Support Vector Machines, the classifier used in this project.
- **Chapter 3** describes the two clustering algorithms investigated in this project (Single Pass and Snob) and points out the issues encountered.
- **Chapter 4** describes the cluster features used to augment the training data and how they are used by Support Vector Machines and Transductive Support Vector Machines.
- **Chapter 5** gives a detailed description of the data sets used for testing as well as the evaluation techniques.
- **Chapter 6** presents the results discovered and discusses their implications.
- **Chapters 7 and 8** draw conclusions from the discovered results and give some useful directions for future research in this area.

Chapter 2

Support Vector Machines (SVMs)

Support Vector Machines (SVMs) (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 labelled 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** (Figure 2.1). Note that Figure 2.1 shows the “best case” in text classification, which means a hyperplane is constructed without any classification errors. This, of course, is not always the case.

The SVM decision function is of the form,

$$f(x) = \text{sign}((w \cdot x) + b) \quad (2.1)$$

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 2.1) such that:

$$\min \frac{1}{2} \|w\|^2 + \frac{C}{m} \sum_{i=1}^m \epsilon_i, \quad y_i(w \cdot x_i - b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0 \quad (2.2)$$

for all $i = 1, \dots, m$, where C is a constant and represents the trade-off between the complexity of the solution and the error penalty ϵ_i (Raskutti et al., 2002). Since the margin between two classes is defined as $\frac{2}{\|w\|^2}$, by minimizing the first term in (2.2), 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 2.1), then the penalty terms from equation (2.2) are omitted.

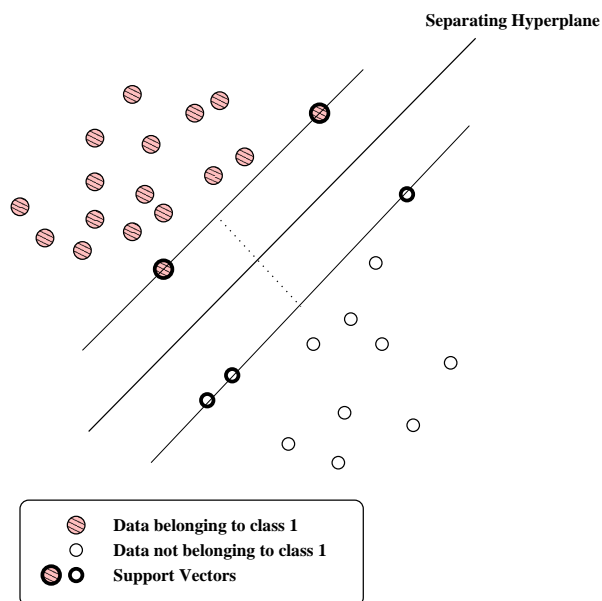


Figure 2.1: SVM solution for two linearly separable classes

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. For more information on Support Vector Machines, see (Schölkopf, 1998; Kwok, 1999; Bennet and Campbell, 2000; Dumais, 1998).

As can be noticed, SVMs use only labelled training data in their learning process. However, we also want to utilize the given set of unlabelled data. We attempt to do that by clustering both the labelled and unlabelled examples and extracting information from the created clusters. The information obtained is then added to the feature vectors in both the training and test set. Then the SVMs are used to learn from that augmented set. This technique enables us to utilize the unlabelled set of documents. The clustering and added features are discussed in more detail in Chapters 3 and 4. The following two sections present an overview

of a few of the most common SVM techniques used in handling multi-class classification problems.

2.1 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.

2.1.1 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 $N \cdot (N - 1)/2$, 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.

2.1.2 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 $N \cdot (N - 1)/2$).

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. A major weakness characteristic for both multi-class SVM techniques mentioned is that they do not cater for the case when a single document belongs to multiple classes. An extension to SVMs which would overcome this limitation is proposed in Chapter 8.

2.2 Transductive Support Vector Machines

Various techniques have been developed for learning to classify documents from a small set of labelled and large set of unlabelled examples (Nigam et al., 2000; Joachims, 1999b; Blum and Mitchell, 1998). This is of great importance since their capability of learning with a small set of labelled examples minimizes the need for manual labeling. The majority of existing techniques tend to label the unlabelled data, as is done by **Transductive Support Vector Machines**, a method introduced by Joachims (1999b). How Transductive Supportive Vector Machines (TSVMs) exploit the given unlabelled document set is explained in the next section.

2.2.1 How does Unlabelled Data Improve Classification?

The key idea behind Transductive Support Machines and the way they utilize unlabelled data is the so called “cluster structure” of text (Joachims, 1999b). As Joachims (1999b) explains, words in documents and in natural language in general, tend to occur in strong co-occurrence patterns. This means that a group of words are very likely to appear together in a document belonging to a certain category, but probably not as likely (or even very unlikely) to appear in a document belonging to a different category. Following a similar discussion given by Joachims (1999b), the example in figure 2.2 shows 6 documents of which D1 and D6 are labelled while the rest of the training set is unlabelled. The unlabelled set of documents still has some very interesting properties. A careful analysis shows that D2 and D3 have subsets of features in common with D1 as well as between each other while D4 and D5 also share “commonality” of feature subsets with D6. This to a reader is not surprising since it is obvious that these two sets of documents most likely belong to the “Sports” and “IT” categories, and both sets have a very specialized structure and terminology. It is this “cluster structure” of documents that is utilized by TSVMs to label the unlabelled documents and create an efficient and accurate classifier function.

	Striker	Goal	Midfield	C++	AI	Hard Drive	the
D1	2	1	1				2
D2		1	1				1
D3		1					1
D4					1		2
D5				2	1		1
D6				1	3	1	1

Figure 2.2: Set of documents showing typical text structure

2.2.2 The TSVM Approach

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” (i.e. unlabelled 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. An example of such a case is given in figure 2.3. The dotted line represents the hyperplane obtained by a SVM on the labelled set, while the solid line represents the final hyperplane obtained by the TSVM after switching the labels of the unlabelled set.

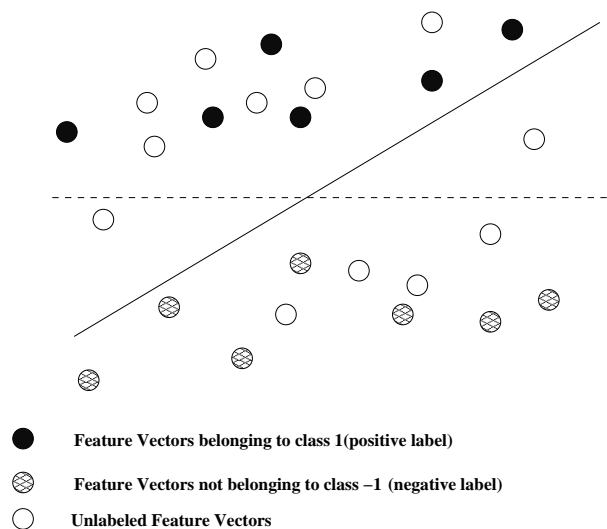


Figure 2.3: Example of TSVM

Advantages: The major advantage of this technique is that it doesn’t require a lot of labelled examples. Experiments performed by Joachims on the Reuters-21578 document collection, and verified using the precision-recall method ¹(Joachims, 1999b), show an improvement in the average of the precision-recall breakeven points from 48.4 for SVMs to 60.8 for TSVMs. The classifying was done using only 17 labelled examples and 3299 unlabelled ones. It should be noted, however, that the same unlabelled set used for training 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’ 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.

TSVM was also used in the testing phase, an approach not very common in verifying the performance of text classification techniques.

Weaknesses: As the project reached its later stages and had started dealing with very large data sets with over 20,000 features it became obvious that the algorithm for TSVMs doesn't always converge. This was the main reason for not continuing with TSVM experiments towards the end of the project. Also, for some data sets TSVMs are very slow, with the decrease in speed compared to SVMs being very big and hard to ignore. 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 labelled 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 labelled examples, which cannot be neglected in the area of text classification. For this reason, the possibility of using TSVMs with feature vectors augmented by different clustering techniques has been investigated. More details on this approach are given in the following two chapters.

Chapter 3

Clustering - Utilizing the Unlabelled Data

Apart from labeling the unlabelled data, which is done by TSVMs, other methods have been proposed for effectively using large amounts of unlabelled data in text classification. The method of particular interest for this project is **clustering**, which is further explained in the following sections.

Clustering is an unsupervised classifying task. In other words, clustering algorithms handle sets of unlabelled data and assign them to groups with the aim of assigning the most similar objects (in our case documents) into the same group. Even though this process doesn't "label" the documents, it gives an idea of how the documents can be separated and which ones can be thought of as possibly belonging to the same category. Hence, clustering can give a "description" of the training set we are dealing with (Fang et al., 2001).

This idea of "describing" the training set is utilized when combining clustering methods and Support Vector Machines (Raskutti et al., 2002). By clustering the documents (both labelled and unlabelled) into similar groups and adding features to each document (Chapter 4) relative to the created clusters, it would be expected that these features would provide extra information to the classifier. Hopefully, that extra information would help discriminate between different categories more accurately. Of course, how helpful these extra features turn out to be depends on how well the clustering algorithm performs or how it discriminates between different categories.

For these reasons, this thesis investigates two clustering methods - the Single-Pass Clustering Algorithm (Raskutti et al., 2002) and Snob (Wallace and Boulton, 1968; Wallace and Dowe, 1994; Patrick, 1991; Oliver, Baxter and Wallace, 1996) and compares their performance when combined with Support Vector Machines and Transductive Support Vector Machines. Also, these combinations are compared to the performance of SVMs and TSVMs alone, in order to reveal whether the added features indeed prove useful to the mentioned classifiers.

3.1 Single Pass Algorithm

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

The distance between document D_i and cluster C_j is calculated as:

$$\text{dist}(D_i, C_j) = \frac{1}{\text{sim}(D_i, C_j)} - 1 \quad (3.1)$$

where the similarity $\text{sim}(D_i, C_j)$ is defined as:

$$\text{sim}(D_i, C_j) = \max\left(\epsilon, \frac{D_i \cdot C_j}{\sqrt{\sum_{i \in D} D_i^2} \cdot \sqrt{\sum_{j \in C} C_j^2}}\right) \quad (3.2)$$

The first term in the *max* function, ϵ , is a small positive number while the second term is defined as the *cosine* similarity between the document and cluster centroid. The cluster centroid is given as the feature vector in which each feature represents the average of the corresponding features in all the feature vectors that belong to the given cluster. Therefore, given a cluster N of documents and all the feature vectors d belonging to N , the centroid C is given as:

$$C = \frac{1}{|N|} \sum_{d \in N} d \quad (3.3)$$

It should be noted that only the training data (including both labelled and unlabelled examples) is clustered. After clustering, new features are calculated and added to the labelled training documents as well as to the test documents. These new features include the document's cosine similarity to the cluster's labelled 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 labelled 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 showed an improvement in the performance of SVMs even if the labelled set is only 0.25% of the total training set. 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 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 predetermined, 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.

One of the main aims of this project was to determine how successful the method of combining SVMs and the Single Pass algorithm is when applied to different data sets and how it compares to other clustering methods, such as Snob.

3.1.1 Implementation

One of the aims of the project was to implement the above described algorithm so that the performance of SMVs and TSVMs with and without these added cluster features could be compared. In the implementation phase, a few issues had to be considered. Following the guidelines provided by Raskutti et al. (2002), it had to be ensured that the clusters were compact and well separated from each other. According to Raskutti et al. (2002), this is achieved when the sum of the following terms is minimized:

1. **Cluster compactness:** the sum of distances of all feature vectors in the cluster to the cluster centroid. The smaller the sum, the more compact the cluster is.
2. **Cluster separation:** the sum of distances between all the cluster centroids and the global centroid. The global centroid is given as the average of all the feature vectors in the training set (Equation 3.3).

Minimizing the sum of these two terms in general provides compact, well separated clusters that will hopefully provide useful features to the classifiers to be used (SVMs and TSVMs). In the implementation phase, the minimum of this sum was found by implementing the Golden Search algorithm as described by Press (1992) which finds the minimum of a given function. The function which minimum was found in this case was, of course, the above mentioned sum. For details about the Golden Search algorithm, and the general implementation in C, see (Press, 1992).

The clustering algorithm was developed using the C programming language and by following the description given by Raskutti et al. (2002). Problems arising during this phase were related to the large memory requirements for the data sets, since clustering was done for thousands of feature vectors with thousands of features. An additional issue is the one of a fairly slow running time, a problem which could maybe even in some cases outweigh any benefits provided by clustering. The problem with the memory requirements was somewhat reduced by using the same representation for the sparse vectors that is used by Joachims in his *SVM^{Light}* application (Joachims, 1999a). That means feature vectors were represented as a series of tuples of the format (featureNo, featureValue) with the features with value 0 being ignored. However, the problem of slow execution remained and is unlikely to be improved upon since the time complexity of the algorithm is $O(N^2)$ (Raskutti et al., 2002).

Due to not being able to obtain the data sets used by Raskutti et al. (2002) until the later phases of the project, the clustering algorithm was developed with a subtle difference compared to the original version. While the original implementation included centroids for each category involved in the clustering (or at least the most populous categories from the training set), the version implemented for this project included centroids only for the “positive” and “negative” class. In other words, all categories apart from the positive one were grouped into the negative class. This, however, opened the possibility of exploring the typical binary classification case - the case when we are given a set of training documents which consists only of “positive” and “negative” documents, disregarding the actual categories to which the negative examples belong.

3.2 Snob

Snob (Wallace and Boulton, 1968; 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:

$$Message_{length}(H, D) = Message_{length}(H) + Message_{length}(D|H) \quad (3.4)$$

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)¹. 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 didn’t seem to be any research on the application of Snob to document classification, it was part of the investigation for this project. The reasons are the impressive results Snob has shown in other classification problems (Dowe et al., 1996; Kissane et al., 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.

The main aim of the project regarding Snob was to determine how the performance of SVMs with added cluster features derived from Snob compares to other implementations (SVMs alone and SVMs with cluster features derived from the Single Pass algorithm). For more detailed information on Snob see (Patrick, 1991; Wallace and Dowe, 1994; Wallace and Dowe, 1996).

3.2.1 Snob in Text Classification

As mentioned earlier, the use of Snob was considered to compare it with the performance of the Single-Pass clustering algorithm when combined with SVMs and TSVMs. The version of Snob used was the Fortran version developed by Chris Wallace (Wallace and Boulton, 1968) since it had a thoroughly written documentation which provided a better understanding of the way Snob works and the different features it offers. The other reason was the realization that the existing C version might have bugs that could affect the results.

The idea was to cluster the documents with Snob using the same data sets as for the Single-Pass algorithm and then add features relative to the clusters created by Snob. Then the performance of SVMs and TSVMs on these augmented training and test docs was evaluated.

¹<http://www.csse.monash.edu.au/~lloyd/tildeMML/>

Using Snob as the clustering algorithm for this particular domain proved to have several difficulties:

- *Number of features:* the number of features used by Snob in the Fortran version is restricted to around 199. The documentation states that increasing this number is possible by changing a parameter and re-compiling all the modules, however this did not work as the files could not compile when the parameter was changed.
- *Feature Distribution:* as input, Snob requires that for each attribute the distribution of that particular attribute is given. Snob handles the following distributions: normal, multinomial, Poisson and von Misses. However, analyzing the data from the TF-IDF Reuters data set (see Section 5.2) showed that the distribution of features resembles a mixture of distributions because usually a large number of documents from the training set has the value 0 for an attribute. In other words, the actual distribution could not be considered as any of the ones handled by Snob. A typical distribution is shown in Figure 14. This implied that pre-processing the data should be considered.

The problem regarding the number of features required being very selective in the process of determining the 199 features (out of tens of thousands of features) to be used when running Snob. In other words, it was important to find the 199 features that discriminate best between different categories in the training set. Following the suggestion of my supervisor, these features were obtained in the following way:

1. SVM^{Light} (Joachims, 1999a) was run 30 times with different splits of the training set. Each run created a model file which has the values for the support vectors and therefore the overall w vector.
2. The w vectors were added in such a way that the absolute values of corresponding features were added. This was done in order to avoid negative values decreasing the total value for a feature. Adding all the w vectors created one w_{total} vector.
3. The 199 features from the w_{total} vector with the largest value were taken as the most discriminative features and used for running Snob.

The reasoning behind this method is that Support Vector Machines, in a simplified explanation, use the w vector as the “discriminator” between categories so it would be reasonable to expect that the features in that vector with the highest value have the biggest influence on discriminating between categories. However, this by no means is the only way of determining the “best” features and other methods could produce better results. This method also raises the question of whether the best features for SVMs are also the best for Snob, since these two methods work and use their features in very different ways, with Snob paying more attention to the distribution of the features.

The distribution of features was an additional problem when applying Snob to this particular domain. As mentioned earlier, the distribution of features on one of the data sets that used the Term Frequency - Inverse Document Frequency (TF-IDF) feature weighting method (for more information, see section 5.2) seemed to be a mixture of distributions since a large number of documents had value 0 for each feature with only a small subset having non-zero values. Since Snob doesn't handle such distributions, it was required to in some way transform the data into a format that could be effectively used by Snob. After careful consideration, the following transformations were tested (and Snob runs named accordingly):

1. **SnobMissingValue:** Using Snob's "missing value" option: Features with value 0 were represented in the Snob file as "missing values" while non-zero values were represented as the log of the actual value (with the intention of normalizing the distribution of the non-zero values). The log function was chosen after testing several options as described in (Box, Hunter and Hunter, 1978). Figures 15 and 16 show the normal probability plots for the non-zero values of a certain feature from the training set before and after the transformation respectively.
2. **SnobDoubleFeature:** Representing features as a combination of two features: As a suggestion from my supervisor, each feature was represented as two features. If the feature value was 0, it was represented with with two zeros - one indicating it isn't a normal distribution and the other for the actual value. If the value was non-zero, it was represented with a 1, indicating it is a normal distribution, and the actual value. As is obvious from this explanation, one of the features is a "boolean" indicating whether the distribution is normal and in the Snob file this feature was given as a discrete variable. The disadvantages of this approach are that the number of features that get used in the end are cut down by half (99 instead of 199) since each feature takes two attributes to describe it. The other problem with this approach is that it is unlikely that Snob can utilize this information correctly since it assumes no correlation between features.
3. **SnobBinary:** Representing features as binary values: This approach was relatively simple and required representing features as 1's or 0's depending on whether their actual value was non-zero or zero.
4. **SnobDiscrete:** Representing features as discrete values: This approach required simply dividing the range of values taken by features into intervals and then giving each feature a value depending on the interval to which their original value belonged.

The next chapter describes the process of augmenting the training set, i.e. the process of adding new features to documents depending on the clusters created by the above mentioned clustering algorithms.

Chapter 4

Adding Features Relative to Clusters

After clustering the training set, usually a large number of clusters is created. The additional features are added relative to the N most populous clusters, where N is some pre-determined number. Unfortunately, up to now there is no method for determining the “best” N . Following the description given by Raskutti et al. (2002), N was chosen to be 20, since that seemed to give the best results for the tested document collections as reported by Raskutti et al. (2002). In the early stages of the project, N was given the value 25 which gave mixed results, slightly improving for some data splits but reducing the performance for others. For this reason it was chosen that the top 20 clusters would be considered, even though further research should be done here.

The next two sections describe the features added to the training set by Raskutti et al. (2002) as well as a set of features tested during the course of this project.

4.1 Features Describing the Training Set

As proposed by Raskutti et al. (2002), after clustering the document collection (the training set only) and choosing the top N clusters, each document is augmented by adding features to it relative to the created clusters. The features added by Raskutti et al. (2002) are:

1. *Closest cluster binary indicator*: This is a binary feature indicating if cluster i is the closest of the N clusters. Therefore, N features are added with all but one having the value 0.
2. *Cluster similarity*: This feature represents the “cosine” similarity of the document to each of the N cluster centroids, therefore adding another N features.

3. *Unlabelled similarity*: This feature indicates the similarity of the document to each of the N unlabelled centroids, i.e. centroids created only from the unlabelled documents in the cluster.
4. *l-centroid similarity*: This feature represents the similarity of the document to each of the centroids in the cluster that are created from documents belonging to the same category. For example, a cluster that has documents belonging to the “sports”, “IT” and “health” categories, will have three centroids (apart from the overall cluster centroid and the unlabelled centroid).

Original Features						Closest Cluster Indicator	Cluster Centroid Similarities			Unlabelled Centroid Similarities			Positive Centroid Similarities		Negative Centroid Similarities					
3	0	2	2	1	6	1	0	0	.21	.45	.01	.13	.46	.09	.45	.78	.38	.11	.08	.17

Figure 4.1: The format of an augmented feature vector

It should be noted, however, that as a result of obtaining the same data sets as Raskutti et al. (2002) in the later stages of the project, the *l-centroid similarity* features were reduced to a 2-centroid similarity, where one centroid was the positive centroid (consisting of documents belonging to the class for which we intend to train the SVM) and the negative centroid consisting of all the other labelled documents, disregarding the category to which they actually belong. As mentioned earlier, even though it is slightly different compared to the original set of features, this method allowed the exploration of this technique when used in the typical “binary classification” situation.

Figure 4.1 shows a simple example of an augmented feature vector. Considering the features were added relative to 3 clusters, we can see that the first three values are binary values. The first “1” indicates that the first cluster out of the three is the closest, hence the other two features get value 0. Next comes the similarity to the different centroids. The first value is the similarity to the first centroid, second the similarity to the second centroid etc. The similarities to the positive and negative centroids are added the same way.

4.2 A New Set of Features

A different set of features was investigated during this project with the intention of discovering how useful different features can actually be. Initial results with a data set using TF-IDF weights (see section 5.2) showed that the performance of SVMs with cluster features as described in the previous section actually degraded. A characteristic of data sets with TF-IDF weights is that the features are usually floating point numbers. As is obvious from the description given in the previous section, the features described by Raskutti et al.

(2002), apart from the similarities which usually are also floating point numbers, include binary indicators for the closest cluster. This feature, of course, is represented by an integer, namely a 1 or 0. The early results led to believe that the type of cluster features used might have to “agree” with the type of the original features. This means that if the original features are represented by floating point numbers, so should the cluster features, in order to avoid “disturbing” the data with unusual values.

For this reason, a slightly different set of features was considered. This new set of features was considered to be used with the TF-IDF data set. It consisted of all of the features described above apart from the binary “closeness” indicator. Also, it had an extra feature that was intended to be used so it can indicate how “positive” a document really is. This extra feature represented the ratio between the similarity of the document to the positive and negative centroid of each of the N clusters. This feature was also a floating point value, which has higher values for feature vectors that are more similar to the positive centroid. Obviously, the limitation of this feature is that it can only be used for clusters which have both positive and negative centroids.

Adding these cluster features, however, did not bring the expected results and therefore did not confirm the hypothesis that the original and added cluster features must “agree” in terms of their types. For this reason, this approach was not further considered.

Another interesting fact observed in the late stages of the project was that the added cluster features don’t have the “cluster structure” typical for text and of importance when classifying documents (described in Section 2.2.1). The question of whether adding cluster features in a “clustered structure” would affect results became interesting. This would mean that feature vectors from one category would have a common subset of features. This already mostly holds in the word frequencies, but it would possibly be of benefit if such a structure existed in the added features. The assumption made here is that feature vectors from the same category would have the same subset of closest clusters, closest unlabelled centroids, closest positive centroids etc. In that case, the new set of features, added relative to N clusters, looks like this:

1. **Add a binary value indicating closest cluster.**
2. **Add the similarity to the closest cluster as one feature (the other $N - 1$ features are set to 0).**
3. **Add the similarity to the closest positive centroid as one feature (the other $N - 1$ features are set to 0).**
4. **Add the similarity to the closest negative centroid as one feature (the other $N - 1$ features are set to 0).**
5. **Add the similarity to the closest unlabelled centroid as one feature (the other $N - 1$ features are set to 0).**

6. For each of the N clusters, if the similarity of the feature vector to the positive centroid is greater than the similarity to the negative centroid, add 1, otherwise add 0.

Such a set of cluster features would produce augmented feature vectors like in Figure 4.2. In this Figure only the added cluster features are shown. Knowing that documents D1 - D3 belong to the one category, as D4 - D7 belong to another, note how the “spread” of these features resembles the structure shown in Figure 2.2. Also, note the difference compared to 4.1 where all augmented feature vectors have the same format of cluster features and the only discriminating factors are their values. Hopefully, the structure shown in Figure 4.2 can help improve the performance of SVMs and TSVMs.

	Closest Cluster			Similarity1			Similarity2			Similarity3		
D1	1			.23				.34		.59		
D2	1			.37				.39			.34	
D3	1				.56			.44		.78		
D4		1				.31	.72					.67
D5		1				.84			.56			.23
D6			1			.78			.98		.45	
D7		1			.67		.34					.83

Figure 4.2: Structure of added cluster features

4.3 SVMs and TSVMs with Cluster Features - Motivation and Methods

As described earlier, one of the main aims of the experiments and the project overall was to determine how useful clustering can be when used for adding features to training documents that are to be incorporated in the learning process of SVMs and TSVMs.

Support Vector Machines, as described in Chapter 2, given a training set of labelled documents, construct a hyperplane by trying to simultaneously maximize the margin between the positive and negative examples and minimize the classification error (Figure 2.1). It is easy to notice that in the training process they make use of labelled examples only. Even

though the method of combining clustering algorithms with SVMs incorporates unlabelled data, the unlabelled data is used to augment the original labelled feature vectors. The training process for the SVM was still executed with labelled examples only. However, the reasoning behind this method is that with these additional features the documents can be described more accurately compared to using only the original features. If this was to hold, SVMs would be able to learn more accurately and perform better with less labelled examples in the training process, at the cost of a usually small increase in the number of features.

On the other hand, Transductive Support Vector Machines in the training phase construct a hyperplane by creating an initial hyperplane relative to the given labelled examples, and then “modifying” it by switching the labels of unlabelled documents until the margin between the two sets is maximized. Therefore, TSVMs include both labelled and unlabelled examples in the training phase.

Having that in mind, the clustering algorithms described earlier were used to augment not only the labelled documents from the training set as described by Raskutti et al. (2002), but the unlabelled ones as well. This allowed the use of the whole training set (including both labelled and unlabelled examples) instead of only the labelled subset as is done with Support Vector Machines.

The difficulties encountered with the use of TSVMs were a very long running time since it was dealing with thousands of unlabelled documents, and as noted in section 2.2.2, the number of possible label switches increases exponentially with the increase of the number of unlabelled documents. This problem amplified with the use of the data set investigated by Raskutti et al. (2002) (see section 5.2), since that training set had 9603 documents and the number of features was over 20,000. For this data set the SVM^{light} application failed to converge, which limited the investigation of TSVMs to only one data set.

Figure 4.3 shows an example of clustering the training set (both labelled and unlabelled examples). The feature vectors get augmented with values depending on the clusters. For simplicity, only one augmented feature vector is shown even though all the labelled feature vectors go through this process (when wanting to use a TSVM, the unlabelled examples get augmented as well). In the picture, the decimal values can be considered the similarities while the binary values are the actual “closeness indicators”. After having the cluster features added, the vectors are classified by a Support Vector Machine (or TSVM).

The next section describes the aim of the experiments conducted, the data sets used for testing the performance of the various algorithms described earlier as well as the various methods used for evaluating the performance of these techniques.

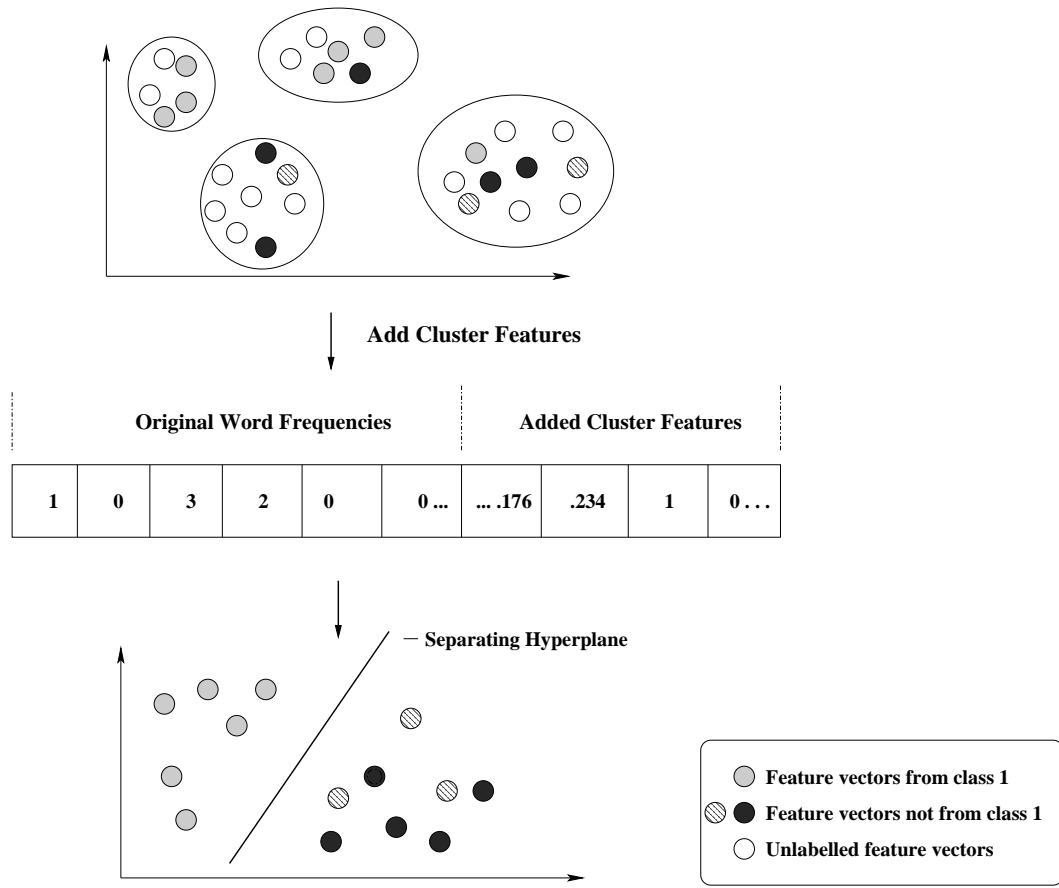


Figure 4.3: SVM using augmented feature vectors as training data

Chapter 5

Experiment Design

5.1 Aim

The use of different data sets and clustering techniques (as well as their variations) brought up many questions during the project phase. This chapter describes the data sets used to evaluate the performance. Also, an overview of the evaluation techniques is given, since there exist many different ones used in this particular domain. The main aim of the experiment was to investigate the various questions arising when using the above described text classification techniques:

1. The performance of this method on data sets very different in size and type of feature weight used.
2. The performance of SVMs when dealing with a training set augmented with features derived by the Single Pass algorithm.
3. The accuracy of this method when documents are augmented with features when Snob is used as the clustering method instead of the single-pass algorithm.
4. The potential of combining the clustering methods with Transductive Support Vector Machines, a method that would also allow the use of unlabelled examples in the learning process.
5. Other potential factors affecting the performance of the described methods.

The next section describes the data sets used for testing the various text classification techniques during the course of this project.

5.2 Data Sets

The first data set used during the course of this project was a data set obtained from the web site of Thorsten Joachims¹. It is a data set consisting of 2000 documents in the training set and 600 documents in the test set and created for the specific task of learning which articles from the Reuters collection are about “corporate acquisitions”. Positive and negative documents were represented in equal numbers which means the training set had 1000 articles belonging to the “corporate acquisitions” category and 1000 which don’t. The same ratio holds for the test set which has 300 positive and negative examples. The total number of features in each feature vector is 9947 with the features in the documents represented using the Term Frequency - Inverse Document Frequency (TF-IDF) scores. The TF-IDF value of a word w in a document d is defined as:

$$\text{TF-IDF}(w) = \text{TF}(w) \times \log \frac{|D|}{\text{DF}(w)} \quad (5.1)$$

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. In other words, this representation considers the frequency of a word not only in a single document but also in the entire document collection. This document collection had the problem of feature distributions that had to be somehow handled by Snob (see Section 3.2.1).

The second data set tested was the modApte split of the Reuters-21578 collection, kindly provided by Raskutti et al. (2002) from the Telstra Research Labs. It consists of 9603 training documents and 3299 test documents where the words were represented using a simple term-frequency technique, which means each feature was given a value indicating how many times it appears in a given document. Feature vectors in this training set had 20,197 features. This number is significantly larger than the one for Joachim’s data set and the representation is also very different, which makes these two data sets suitable for investigating the effect of the experiments on different document collections. For the modApte split, the second most populous category was chosen as the “positive” category (1200 examples), with the rest being considered as negatively labelled data. Note that this set did not have the problem of having its feature distributions represented in Snob. The features were input to Snob as two kinds of values: binary and discrete, which was possible since the feature values were simple integers. Both data sets had word stems while the stopwords were removed. Stopwords are words with almost no discrimination power since they appear in almost any category of documents (such as “and”, “the” etc.).

For both data sets, the investigated techniques were tested with having 1%, 5% and 10% labelled data out of the whole training set. Each of these proportions were tested with 10 random splits and an average was obtained. Also, following the testing conducted by

¹URL: <http://svmlight.joachims.org/>

Raskutti et al. (2002), it was ensured that the proportion of the total number of labelled examples is equal to the proportion of positive examples compared to the subset of documents belonging to the positive class. In other words, if the document set had 9603 documents and the positive class had 1000 documents in that collection, then a 10% split would have 100 positive examples, approximately 860 negative examples, and the rest would be unlabelled.

5.3 Evaluation

A common technique for evaluating the performance of classifiers is the *confusion matrix* as shown in Table 5.3 (Kohavi and Provost, 1998). It contains 4 values:

	Predicted Negative	Predicted Positive
Actual Negative	A	B
Actual Positive	C	D

Table 5.3: The confusion matrix

- A is the number of correct predictions that a document belongs to the negative class
- B is the number of incorrect predictions that a document belongs to the positive class
- C is the number of incorrect predictions that a document belongs to the negative class
- D is the number of correct predictions that a document belongs to the positive class

These numbers on their own provide a lot of information without telling us much. That is why they are usually combined into other, more useful measures:

1. **Accuracy** (Sebastiani, 2002) is the percentage of correct decisions made by the classifier on the given test set. Accuracy is determined from the following equation:

$$\text{Accuracy} = \frac{A + D}{A + B + C + D} \quad (5.2)$$

2. **Probabilistic scoring** (Needham and Dowe, 2001) determines how effective a classifier is by estimating the number of bits required to describe the test set given the structure it has after being classified by a particular technique. To give the equation for probabilistic scores, we first need to define a few other terms:

- The recall or number of true positives (TP) is the proportion of positive documents that were recognized by the classifier as being positive:

$$\text{TP} = \frac{D}{C + D} \quad (5.3)$$

- The number of false positives (FP) is the number of negative documents that were recognized as positive by the classifier:

$$FP = \frac{B}{A + B} \quad (5.4)$$

- The number of true negatives (TN) is the number of negative documents that were recognized as negative by the classifier:

$$TN = \frac{A}{A + B} \quad (5.5)$$

- The number of false negatives (FN) is the number of positive documents that were recognized as negative by the classifier:

$$FN = \frac{C}{C + D} \quad (5.6)$$

From these formulas we can calculate two parameters p and q as follows:

$$p = \frac{TP + 1}{TP + FP + 2}, \quad q = \frac{TN + 1}{TN + FN + 2} \quad (5.7)$$

Finally, the probabilistic score is calculated from the following equation:

$$\text{ProbScore} = -TP \cdot \log_2 p - FP \cdot \log_2 (1 - p) - TN \cdot \log_2 q - FN \cdot \log_2 (1 - q) \quad (5.8)$$

These two measures have one important difference that should be noted before reading the Results section (Chapter 6): the better the performance of the classifier, the higher is the accuracy - and the lower is the probabilistic score. Accuracy and probabilistic scoring seemed to give very similar results, in the sense that classifiers with the highest accuracy ended up having the lowest probabilistic score. The probabilistic score, however, proved useful, since it was capable of differentiating better between classifiers with similar accuracies.

5.4 Partitioning the Data

As part of the experiment, in the later stages of the project, one of the larger training sets was split into 5 equally sized partitions. Each partition was clustered separately and features were added relative to the clusters from all partitions. This, of course, increased the number of added cluster features 5 times. The motivation for this was to reduce the computational and memory requirements associated with the Single Pass algorithm (Raskutti et al., 2002) but also to test whether an increased number of cluster features has an influence on the performance of Support Vector Machines. This approach was also tested by Raskutti

et al. (2002), who reported that the number of partitions didn't have a serious impact on results and that partitioning should be considered solely for resolving computational issues. However, in the project a slightly different approach was taken, since partitioning was done with the condition that each partition needed to have an approximately equal number of labelled examples. This was done to ensure that in each partition at least a few clusters with labelled centroids are obtained. Partitioning the data this way gave very interesting results which are reported in the following chapter.

Chapter 6

Results

6.1 SVM vs SVM+Cluster

The aim of this section is to present and analyze the results obtained by comparing the performance of the following techniques:

1. Support Vector Machines and Transductive Support Vector Machines alone
2. SVMs and TSVMs with added cluster features.
3. SVMs with added cluster features with the note that clustering was done on partitions of the data set rather than the whole data set, as explained in section 5.4

It should be also noted that the results for TSVMs are only given for the TF-IDF Reuters data set because of the problem of SVM^{light} (Joachims, 1999a) not converging when dealing with the larger data set. The notation used for the results is as follows:

- **SVM and TSVM**: Results obtained by applying the SVM or TSVM alone
- **SVM + SinglePass and TSVM + SinglePass**: Results obtained by applying the SVM and TSVM to a data set with added cluster features derived from the Single Pass algorithm
- **SVM + SnobBinary, SVM + SnobDiscrete, SVM + SnobMissingValue, SVM + SnobDoubleFeature**: Results obtained by applying the SVM and TSVM to a data set with added cluster features derived from Snob with different ways of handling the distributions. The same format is used for TSVMs when combined with Snob.

- **Partition:** Any of the above notations (apart from the first one) combined with “**Partition**” denote that the results were obtained with added cluster features that were derived by first partitioning the data set and then clustering those partitions.

The results shown in Figures 6.1, 6.2 and 6.3 show the performance of all the techniques (SVM, SVM + SinglePass, SVM + Partition + SinglePass, SVM + Snob, SVM + Partition + Snob) on the TF-IDF data set for splits of 1%, 5% and 10% of labelled data respectively. Note that only the best performing “version” of Snob was chosen out of the 4 for this display. The results are displayed as probabilistic scores. For the accuracy scores, see Figures 2, 3 and 4 in the Appendix. The results show that, contrary to expectations, adding cluster features actually degrades the performance of SVMs. For all three splits, SVMs alone outperformed SVM + SinglePass and the three Snob versions which performed best for the different splits. However, interesting results were obtained when partitioning the data. From Figures 6.1, 6.2 and 6.3, it is obvious that partitioning the data and then clustering the partitions using the Single Pass method improved compared not only to SVM + SinglePass (i.e. without partitioning) but also compared to SVMs alone. Partitioning the data and then clustering with Snob also showed improvement compared to using Snob without partitioning, however it still did not outperform SVMs without added cluster features. Figures 6.1, 6.2 and 6.3 also confirm the value of labelled data since we can see that with an increased proportion of labelled feature vectors the performance improves as well. The value of labelled data was noticed in the earlier stages of the project and is one of the reasons for ensuring that when partitioning each partition had a certain number of labelled feature vectors (section 5.4).

Figures 6.4, 6.5, and 6.6 show the results for the same data set but this time when TSVMs are combined with clustering. In this case it is obvious that neither of the clustering methods succeeded in enhancing the performance of the TSVM, which on it’s own was already performing exceptionally well, probably due to the fact that it had a large number of training examples (2000). This, of course, was because TSVMs use both the labelled and unlabelled data in their training process. An interesting observation that the results obtained with Snob in some cases are better than those obtained with the Single Pass algorithm. However, overall neither of these methods outperformed TSVM alone, which diminished the value of such observations. Again, the results here are displayed as probabilistic scores, for the accuracies see Figures 8, 9 and 10 in the Appendix.

Figures 6.7, 6.8, and 6.9 show the probabilistic scores for the 4 techniques (SVM, SVM + SinglePass, SVM + Partition + SinglePass, SVM + Snob, SVM + Partition + Snob) when applied to the larger word-frequency Reuters data set. These results show again that SVMs with cluster features derived by partitioning the data and clustering with the Single Pass algorithm outperforms the other techniques. However, an interesting fact is that for the data set with 10% labelled data (Figure 6.9), SVMs without cluster features performed best, indicating that the story doesn’t really end with partitioning and that there are other factors influencing the performance of these methods. Again, SVMs with Snob features seemed to struggle compared to the other methods. For the accuracy scores, see Figures 5, 6 and 7.

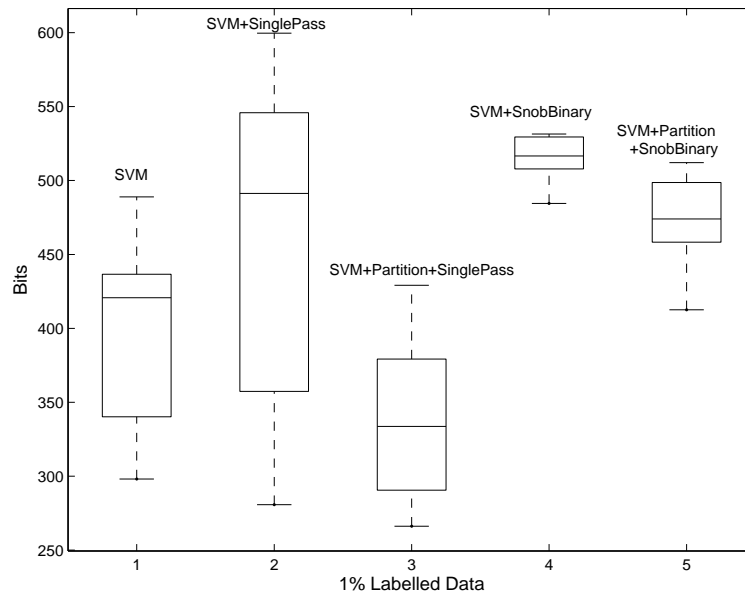


Figure 6.1: SVM probabilistic scores for 1% labelled data on TF-IDF Reuters

Probably the most interesting results were obtained when testing SVMs with training data augmented with the new features as described in Section 4.2. Since this was an idea that was observed in the very late stages of the project, the tests were conducted only for the SVMs on the TF-IDF Reuters set (without partitioning). The time needed for setting up the experiments and the execution time of the clustering algorithms (which are quite slow) did not allow quick runs and gathering of results. Snob was not considered anymore because of the previous results, as were not TSVMs. Note that the SVM + SinglePass approach with new cluster features was compared to the SVM alone and SVM + SinglePass with old features approaches (with no partitioning). The results obtained are very interesting. Figures 6.10 and 6.11 show that the new features performed better than both SVMs alone and SVMs with the old cluster features. Figure 6.12 shows a slightly lower value of the probabilistic score for the SVM with new features compared to SVMs alone. However, the accuracy, given in Figure 13 shows that the SVM + SinglePass method with new features has a slightly higher accuracy (this was also the first time that the probabilistic and accuracy scores didn't agree). For the accuracy scores, see Figures 11, 12 and 13 in the Appendix.

6.2 Discussion

Cluster structure of documents: The results from the experiments suggest that Support Vector Machines are sensitive to added features and that in general they perform better

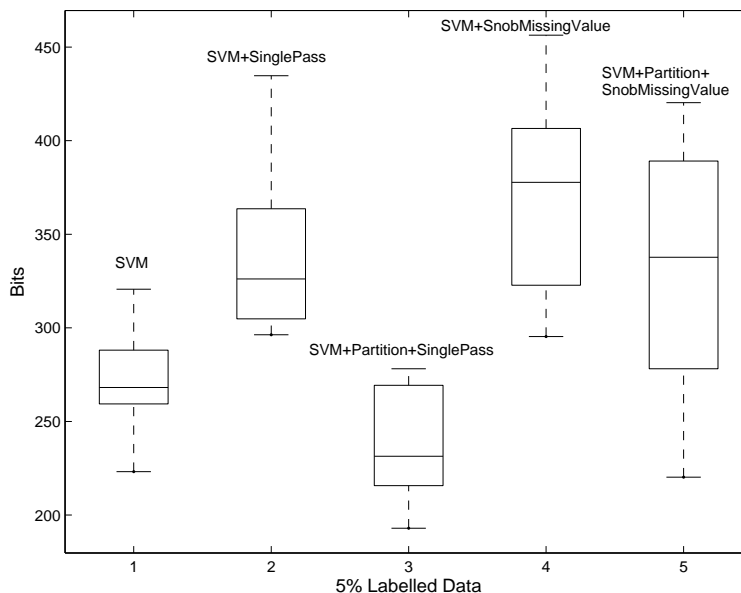


Figure 6.2: SVM probabilistic scores for 5% labelled data on TF-IDF Reuters

without added cluster features rather than with them. A probable explanation for this is that the features they are dealing with when combined with clustering methods (i.e. the original word frequencies and the cluster features) are different in nature, with different distributions and different meanings. Also, while words tend to be “clustered” within one category (see Section 2.2.1), which means a certain group of words will appear only in one category of documents, the added similarities to cluster centroids don’t have that typical “distribution”. The added features are spread among documents of all categories, with the only discriminating factor being the actual values of those added features, not their “spread” among categories. This was also confirmed with the results obtained when adding the new set of features which were given such a “cluster structure”. SVMs seemed to differentiate better between categories better when the added features had that same typical structure as the set of original features.

The importance of the structure of features is even more apparent in the case of Transductive Support Vector Machines. In Figures 6.4, 6.5, and 6.6 we can see that TSVMs without cluster features performed better than any other method for all 3 splits. TSVMs, as described in Section 1.2.1, take advantage of that typical “cluster” structure of words within the same category which helps them utilize the given unlabelled data. However, when cluster features are added, this structure is lost (at least for the “cluster derived” subset of features). The results suggest that this disruption of structure affects the performance of TSVMs, unfortunately not the way we would prefer.

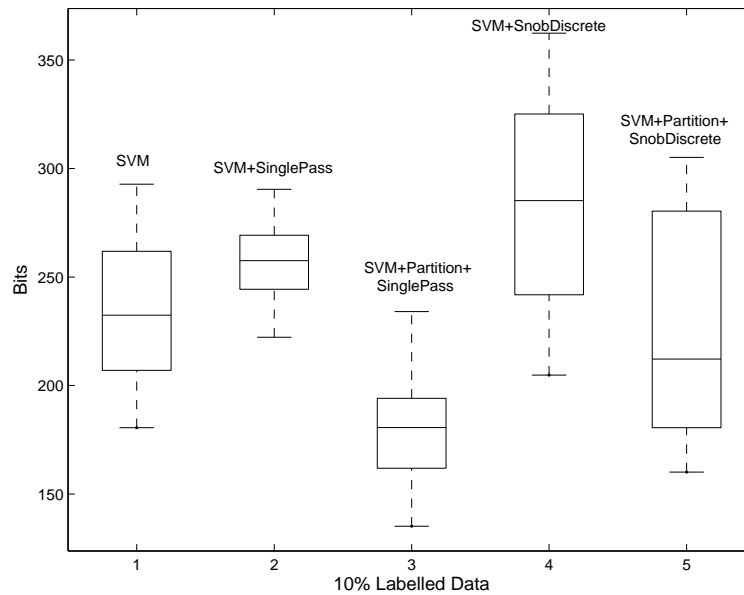


Figure 6.3: SVM probabilistic scores for 10% labelled data on TF-IDF Reuters

Quality of features: Another possible factor is the quality of the added features. Namely, the adding of cluster features is performed with the assumption (or rather hope) that each cluster contains feature vectors belonging to mostly one category, i.e. that they are “pure”. However, if they are not, then the unlabelled centroid, as well as the negative centroid, will consist of feature vectors belonging to different categories. This means that when adding the similarities of these centroids to the feature vectors from the training set as cluster features, the discrimination factor between them will be “blurred” and significantly reduced. For example, if a cluster had a pure unlabelled centroid, where every feature vector belonged to the category “sports”, then the similarity of a “sports” feature vector to that centroid would most likely be high, and any other feature vector would have a low similarity value. These added features then can be considered as having some discrimination power. However, if our unlabelled centroid represents a mixture of different categories, the similarity values for feature vectors from different categories won’t necessarily be as distinct, which reduces their discrimination power.

Partitioning the data: The feature quality factor becomes even more likely when analyzing the results obtained when first partitioning the data and then clustering it using the Single-Pass algorithm. As was seen earlier, this technique performed best in most cases, even when compared with SVMs without cluster features. As a minor test and an attempt to explain this fact, a few runs of the Single-Pass algorithm were run on the TF-IDF data set with all 2000 examples labelled. This was done with and without partitioning. The

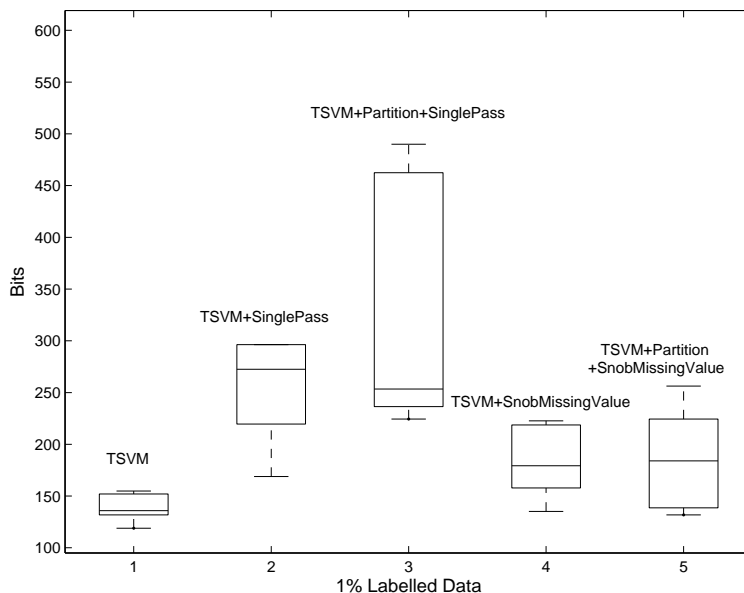


Figure 6.4: TSVM probabilistic scores for 1% labelled data on TF-IDF Reuters

interesting result was that the clusters obtained by clustering the partitions seemed in general to be more “pure” (but still not completely) compared to the clusters obtained by clustering the whole training set. This fact emerges as one of the more likely explanations for the better performance of SVM + Partition + SinglePass and also gives more substance to the theory from the previous paragraph.

Snob: Snob proved to be not as successful in this particular domain. It constantly performed below the level of the other classifiers, with its performance approaching best only on a few occasions. This suggests that Snob didn’t overcome some of the difficulties like the limited number of features and handling the distributions from the TF-IDF Reuters data set. In fact, considering the fact that Snob was dealing with only 2% (199 out of 9947) of the total features from the TF-IDF Reuters set and approximately 1% out of the modApte Reuters split (199 out of 20,197), it could be said that the overall performance of Snob wasn’t that bad. Hopefully, other methods of selecting the 199 features rather than that explained in Section 3.2.1 and other ways of handling the feature distributions could improve results.

Different data sets: The performance of the tested methods showed some differences between the two data sets. Especially comparing figures 3 and 9, we see that SVMs alone performed better than SVM+SinglePass for the modApte Reuters split, as opposed to the results obtained for the 10% split of the TF-IDF Reuter set. This suggests that the performance of SVM + Cluster techniques depends also on the size of the training set as

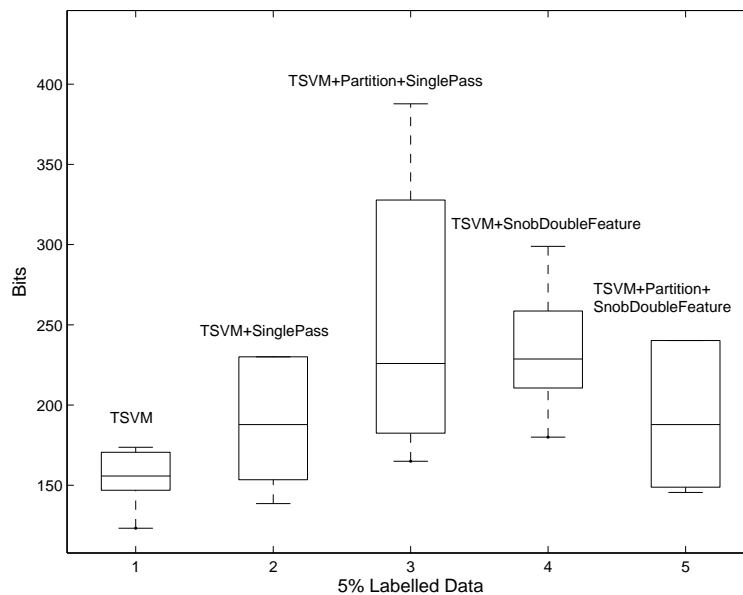


Figure 6.5: TSVM probabilistic scores for 5% labelled data on TF-IDF Reuters

well as the number of unique features. This might also mean that the number of partitions for the modApte split (which was about 5 times larger than the TF-IDF set) should have been different compared to the 5 used for the TF-IDF set. That is possible because it makes sense to use different parameter settings for training sets that are different in nature and size.

The next chapter draws conclusions from the obtained results and the above discussion, and the thesis ends with directions for future work which could be approached in this particular domain.

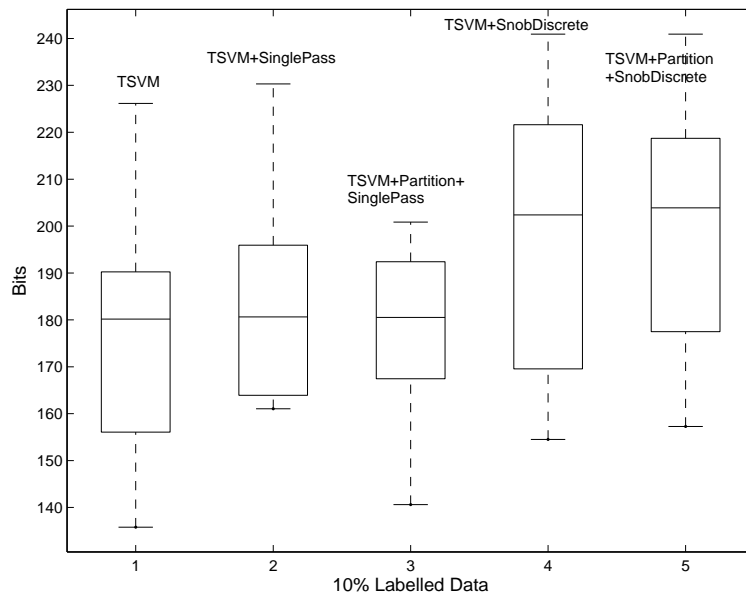


Figure 6.6: TSVM probabilistic scores for 10% labelled data on TF-IDF Reuters

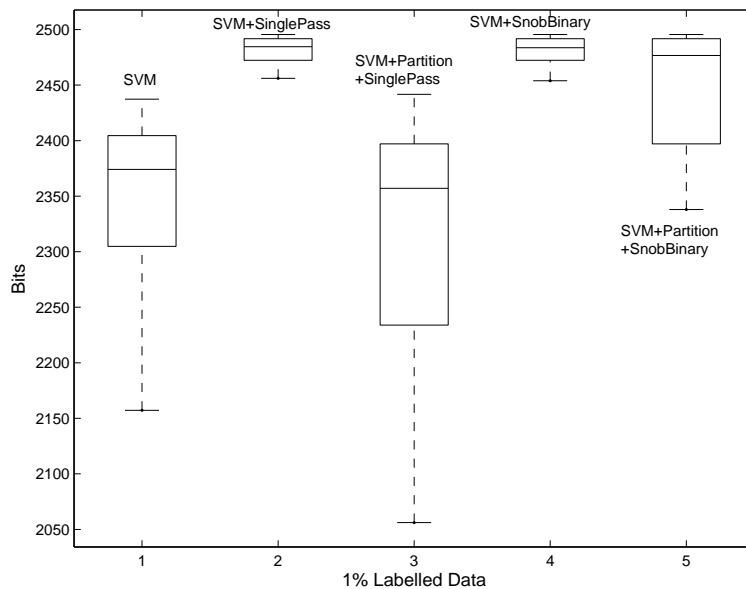


Figure 6.7: SVM probabilistic scores for 1% labelled data on modApte Reuters split

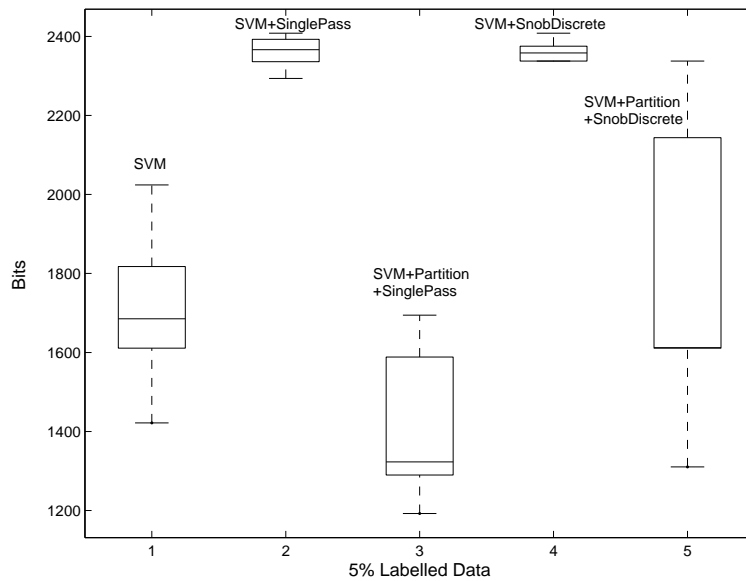


Figure 6.8: SVM probabilistic scores for 5% labelled data on modApte Reuters split

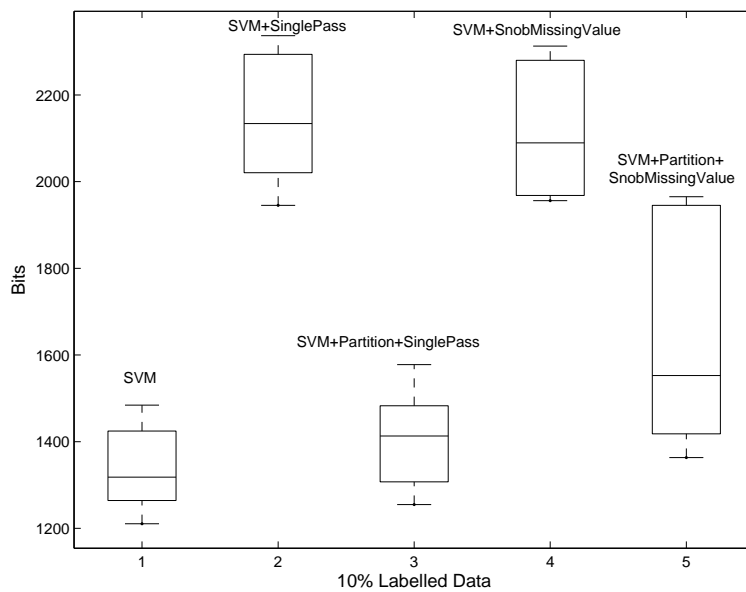


Figure 6.9: SVM probabilistic scores for 10% labelled data on modApte Reuters split

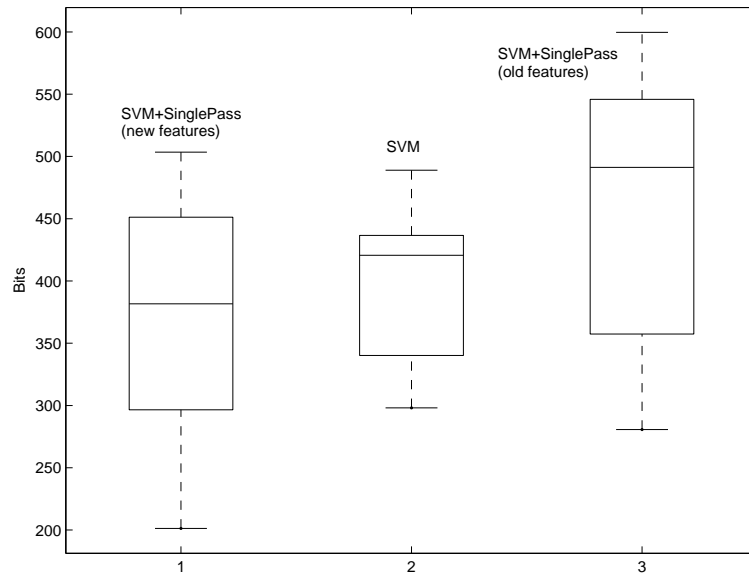


Figure 6.10: SVM probabilistic scores for 1% labelled data on TF-IDF Reuters with new cluster features

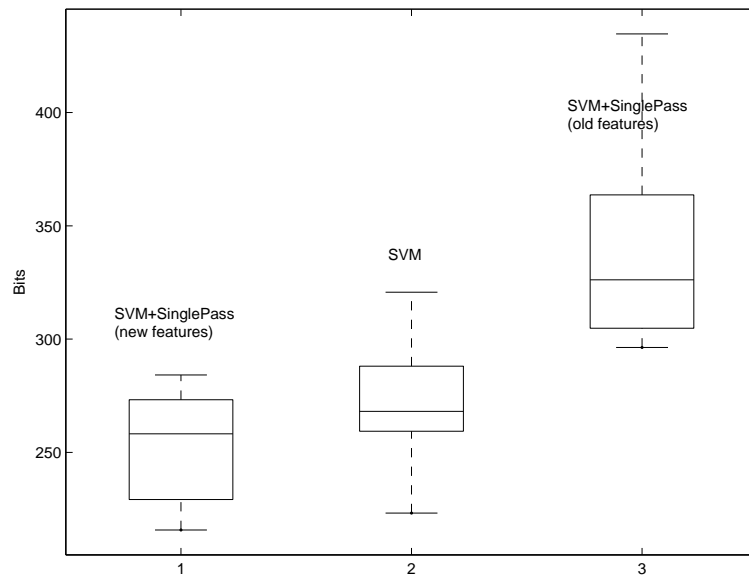


Figure 6.11: SVM probabilistic scores for 5% labelled data on TF-IDF Reuters with new cluster features

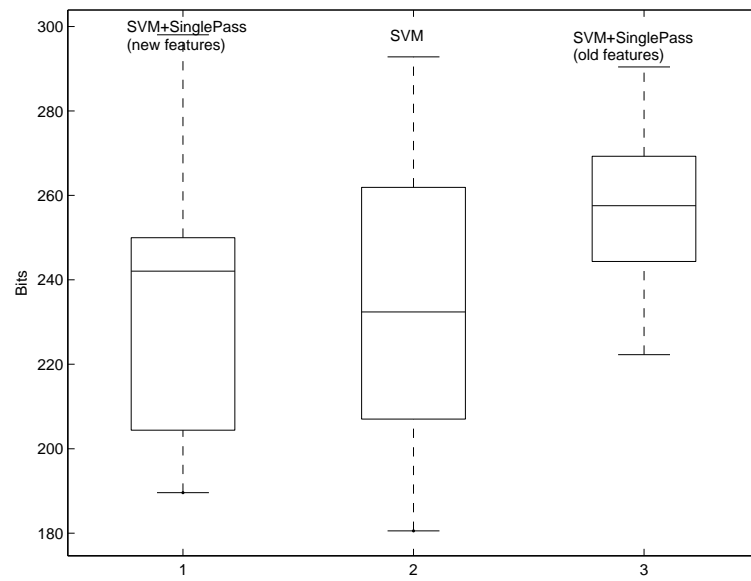


Figure 6.12: SVM probabilistic scores for 10% labelled data on TF-IDF Reuters with new cluster features

Chapter 7

Conclusions

The performances of the tested text classification techniques are interesting because they indicate several facts about Support Vector Machines and especially their application in text classification . The most important conclusions that can be drawn are as follows:

- **Analyzing the data set is of vital importance:** Apart from choosing the appropriate classification technique, it is just as important to analyze and be aware of the training set - for example its size and feature types used. The difference in results between the TF-IDF Reuters data set and the modApte split of the Reuters set indicates this since the same techniques on these sets gave somewhat different results. As discussed in the previous section, this could be because of the difference in the size of the data sets but also because of the difference in the type of features used as well as the number of features. Different data sets might require a different number of partitions, different number of clusters etc. This seems straightforward because it is common sense that a data set of size 200 shouldn't really be partitioned the same way a data set of size 20,000 is. Another vital factor, at least when it comes to text classification, is the "cluster structure" of features, typical for documents. Results, even though preliminary, show that if the added features have the same type of "spread" among categories as the original features, improvements in results can be obtained even without partitioning.
- **SVM's sensitivity to added features:** Support Vector Machines, contrary to belief, are not a technique with which one can simply "throw in" data randomly and expect the SVM to make sense out of it and produce results. Results actually show that SVMs are sensitive to added features, especially features that are not correlated, like in our case word counts and document similarities, which are typically two different types of features. As discussed earlier, added features often also disrupt the "cluster" structure of words within a single category, an important characteristic in text classification and utilized by SVMs in their learning process. When they are added in a "clustered" fashion, results actually improve.

- **Partitioning the data improves SVM performance:** Even with the two previously mentioned points, clustering partitions of the training set and then adding features according to these sets of clusters (instead of one set) still seems to improve the performance of SVMs. As discussed in Section 6.2, one of the reasons for this could be the increased number of cluster features but also the quality of the features since they are derived from clusters that seem more “pure” rather than the original clusters before partitioning. These results, however, should be taken with caution because different data sets give different results and further research should be made in order to generalize the technique for all data sets.
- **TSVM’s limited applicability to text classification:** Even though very successful when given a large enough training set, TSVMs seem to have a problem when the training set becomes too large, both in terms of number of feature vectors and number of unique features. The problem is expressed in their inability to converge for certain data sets (in this project the modApte split). This limits their use to only certain data sets. Also it should be noted that they do not “blend” well when cluster features are added and seem even more sensitive to the disruption of the original data than SVMs.
- **Efficiency of clustering algorithms:** Clustering, as we have seen, can produce features that improve the performance of SVMs if extreme care is taken. However, in real life there is always the issue of efficiency and time. The Single-Pass algorithm, for example, runs extremely slowly, especially because of the search for the optimal threshold T (Section 3.1) which requires it to cluster the training set several times. In situations where the time in which the classification is done is of great importance, the speed (or lack of it) might outweigh the benefits provided by clustering. Snob was significantly faster, however this was most likely because of the small number of features it was dealing with.
- **Snob in text classification:** The use of Snob in text classification is currently limited because of the handling of distributions of features in the TF-IDF set as well as the limited number of features it can deal with. This, however, doesn’t mean that better ways of handling these limitations couldn’t improve current results and it is definitely a direction for future work.

As a final word in this conclusion, it should be noted that this project has really only scratched the surface when it comes to discovering the possibilities and the potential of the explained text classification methods as well as exploring, analyzing and overcoming some of their limitations. It does, however, provide a basis on which future projects can build on and hopefully further improve our understanding of the various methods discussed. The results obtained in the course of this project as well as the difficulties encountered raise many questions and open many possibilities for further research. Some of those possibilities are mentioned and discussed in the next chapter.

Chapter 8

Future Work

Setting the parameters: A characteristic of the text classification methods investigated is that there are a few parameters fixed without regard to the size of the data set, number of features or other characteristics of the data set. Above all, these are the number of clusters considered when adding cluster features as well as the number of partitions. In the experiments described above, the number of clusters taken was $N = 20$ while the number of partitions was $k = 5$. However, there is no reason to believe or formal proof saying that these parameter values are optimal. Indeed, the results obtained with the modApte split of the Reuters data set which was much larger than the TF-IDF set show that these parameters are not optimal. Even though in situations like this it is hard to avoid some predetermined thresholds, for future research it would be useful to investigate methods of determining the optimal values for these parameters depending on the data set and given clusters. The number of clusters to be considered might be variable and depending on some criterion. A few suggested criteria are:

1. Take top N most populous clusters which contain at least $k\%$ of the documents in the training set (or partition). Note that here we still have a threshold to satisfy, but now we are avoiding the possibility of adding features relative to clusters that are so small they can be considered insignificant for describing the document set.
2. Take $\leq N$ most populous "pure" clusters. This means that the only clusters that should be considered are those that, apart from the unlabelled data, contain only one of the positive or negative documents, but not both. The reasoning is that these clusters can be thought of as providing better quality features since their centroids are "pure" (or at least we hope they are, because we don't know the labels of the unlabelled data). Of course, the number of clusters here is again variable because we don't know how many such clusters exist. Note that a similar approach was attempted in the course of the project, where these pure clusters were "merged" in order to increase the size of the labelled centroids. This method, however, did not

give satisfactory results and was not continued. This could have something to do with the fact that after merging the clusters aren't compact and well separated anymore.

3. Information gain (Yang and Pedersen, 1997) could be investigated as a technique for determining the most significant clusters, i.e. those that provide most information. However, this method does suffer from high computational and memory costs which could seriously limit its applicability.

As noted earlier, these methods still will have some thresholds to satisfy, but they can be set at some reasonable values and it makes more sense to fix them, rather than fixing the number of clusters and then taking the risk of adding cluster features that are useless and maybe even harmful to the performance or the SVM.

Better Ways of Dealing with Snob Limitations: One of the major limitations associated with using Snob in text classification was the limited number of features it handles. This was attempted to be resolved in the course of the project by using the method described in Section 3.2.1. However, this might not be the best way of choosing the best 199 features and certain methods that are closer to the way Snob works might be more appropriate. Information gain (Yang and Pedersen, 1997) seems like a likely candidate, but other techniques might be useful as well. Also, it would be useful to investigate other methods for handling the unusual distributions of features from the TF-IDF Reuters data set.

Multi-class Classification: As mentioned at the beginning of this thesis (Chapter 1), it is often useful to be able to indicate the possibility of an object belonging to several classes or categories. This is particularly true for text classification, since documents can belong to several classes. If we go back to the example of the news filter (Chapter 1), we might want our application to leave news articles that have something to do with both "health" and "fitness", even if we don't want to read articles that belong to "fitness" alone. Figure 8.1 gives an example of such a situation, where we can imagine Class 1 as being "health" and Class 2 being "Fitness".

In this section an extension to SVMs is proposed that will handle the case of multi-class text classification. Due to limited time, this technique was not tested. The main idea behind this solution is slightly modifying the output of SVMs. Instead of taking the sign of the decision function as shown in Equation 2.1, the actual value of the function is taken. This value can then be taken as a certain "confidence" score of "how much" does a certain feature vector belong to the given class. The reasoning is that the further the feature vector is from the separating hyperplane (from the "positive" side), the more certain we are it belongs to that class, and hence the greater will be the value of the decision function. The closer it is to the hyperplane, the lower the value of the decision function (it approaches zero) and therefore the lower is our confidence about this feature vector belonging to the given class.

Back to our example, if feature vector D from Figure 8.1 is classified as "positive" by both f_1 and f_2 , then we can obtain the values of $f_1(D)$ and $f_2(D)$ but instead of taking the One vs. Rest approach (Section 1.1.2) and choosing either Class 1 or Class 2 as the valid class for D , the following is done:

1. Calculate the confidence that D belongs to Class 1 as follows:

$$\text{Class1Score}(D) = \frac{f1(D)}{f1(D) + f2(D)} \quad (8.1)$$

2. Calculate the confidence that D belongs to Class 2 as follows:

$$\text{Class2Score}(D) = \frac{f2(D)}{f1(D) + f2(D)} \quad (8.2)$$

This means that the output of our classifier would now give confidence scores rather than just the single class it believes D belongs to. Note that if only one of the classifiers claims D as belonging to their class, then D would be labelled as such. Also, if both $f1(D)$ and $f2(D)$ were negative, then D would be considered as belonging to neither of these classes.

Of course, this approach can be extended to more classes even though there is still the efficiency issue since N different classes require N SVM classifiers. It also should be noted that this suggestion can easily be extended to more sophisticated and possibly more accurate measures of determining the confidence scores. One of them, as a suggestion from my supervisor, could be the actual distances of D from both hyperplanes. Both of these methods, as well as others, could prove to be very useful since when dealing with documents it is important to gracefully handle the case of multi-class documents. Of course, it would be interesting to test one of these methods on augmented feature vectors and see how that affects the confidence scores.

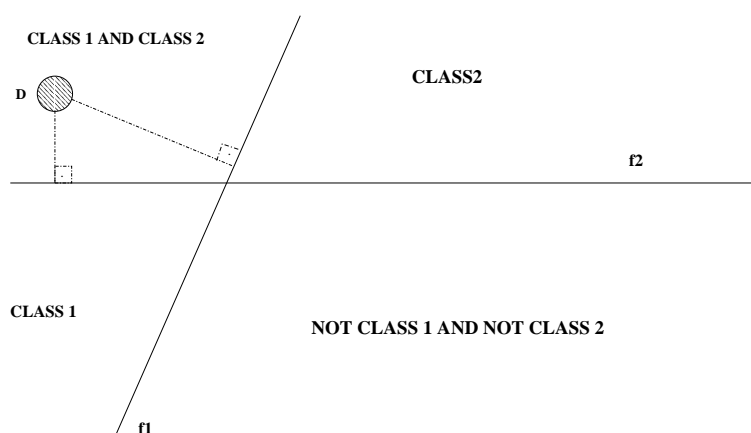


Figure 8.1: Example of a multi-labelled document

“Cluster structure” in added features: As the new set of features (described in Section 4.2), seemed to produce very good results even without partitioning, investigating this

approach would be of great importance and could produce further exciting results. This would also confirm the importance of being careful not to disrupt the structure of the training set with new cluster features. It would be very interesting to see how this approach would perform when combined with partitioning. Hopefully, this could “blend” the benefits of both having quality features derived from “pure” clusters and maintaining the typical “text structure” of the added cluster features.

Accuracy Scores

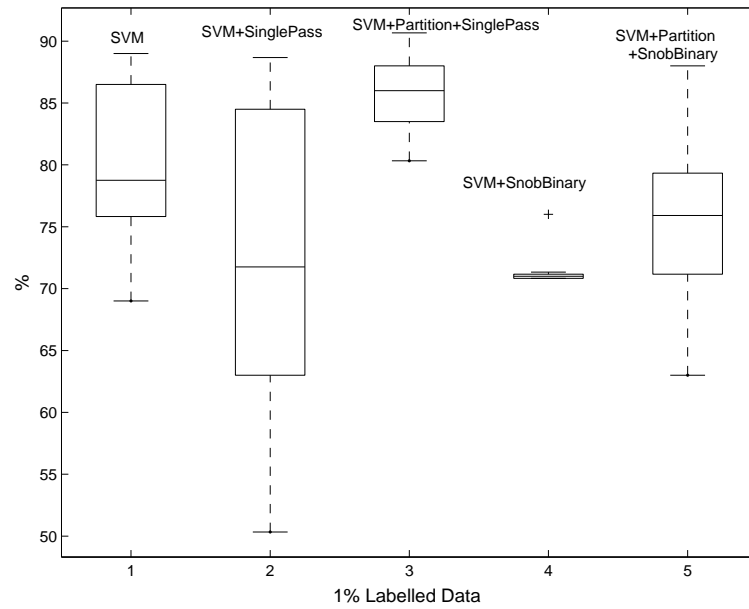


Figure 2: SVM accuracy scores for 1% labelled data on TF-IDF Reuters

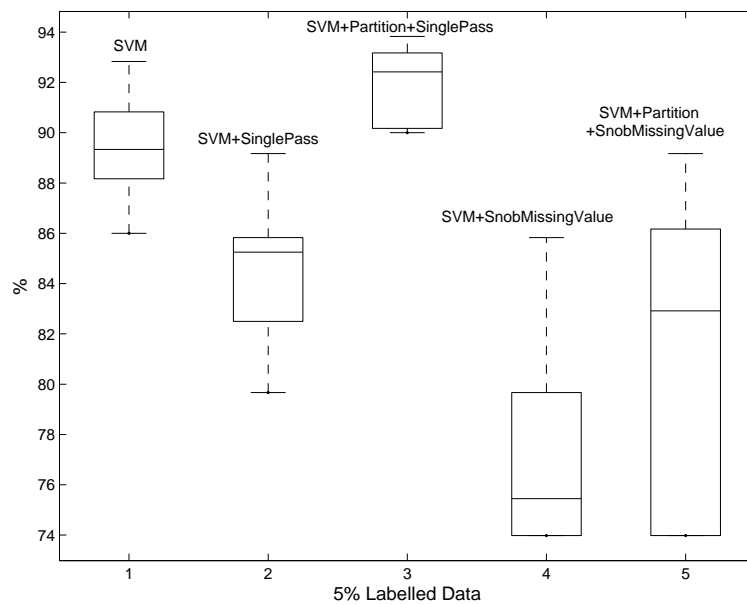


Figure 3: SVM accuracy scores for 5% labelled data on TF-IDF Reuters

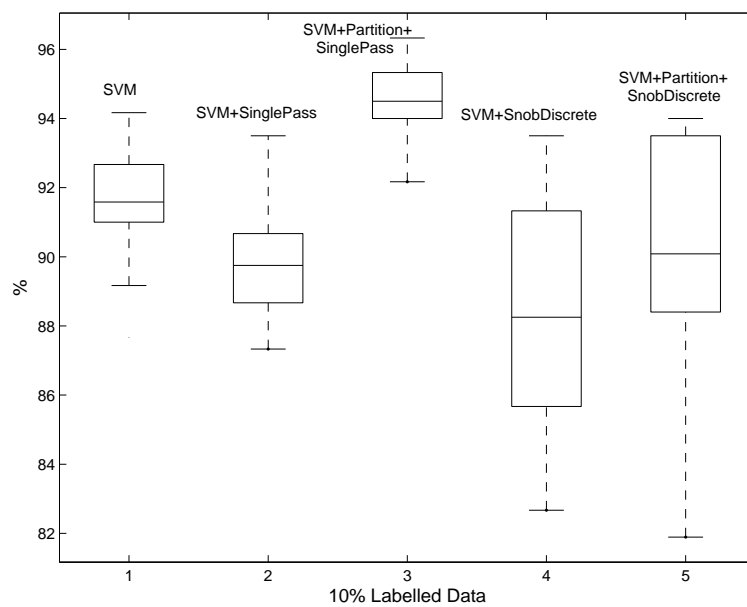


Figure 4: SVM accuracy scores for 10% labelled data on TF-IDF Reuters

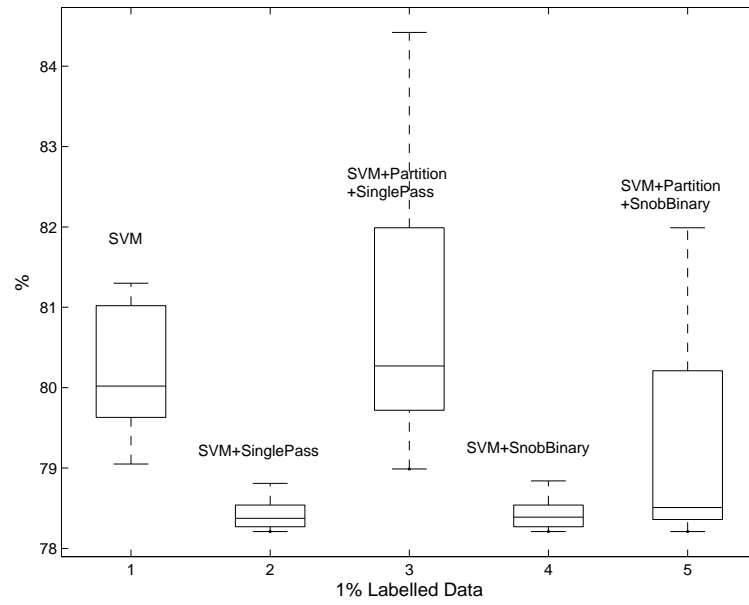


Figure 5: SVM accuracy scores for 1% labelled data on modApte Reuters split

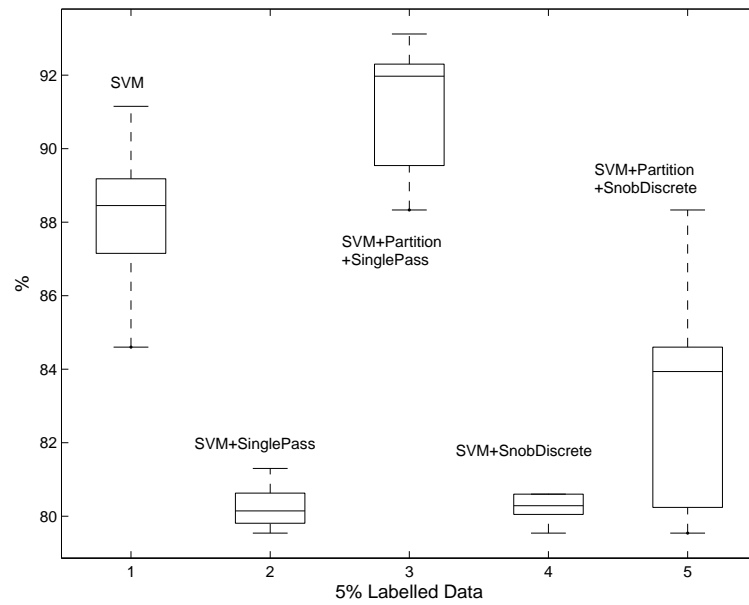


Figure 6: SVM accuracy scores for 5% labelled data on modApte Reuters split

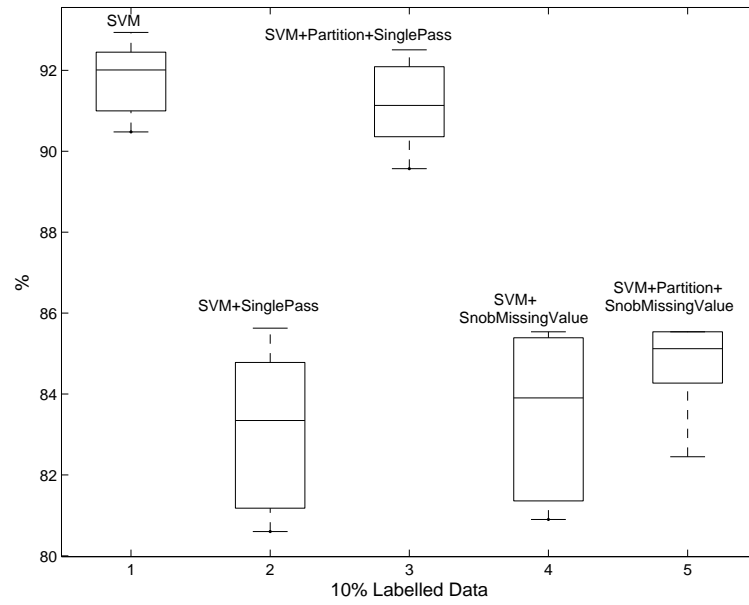


Figure 7: SVM accuracy scores for 10% labelled data on modApte Reuters split

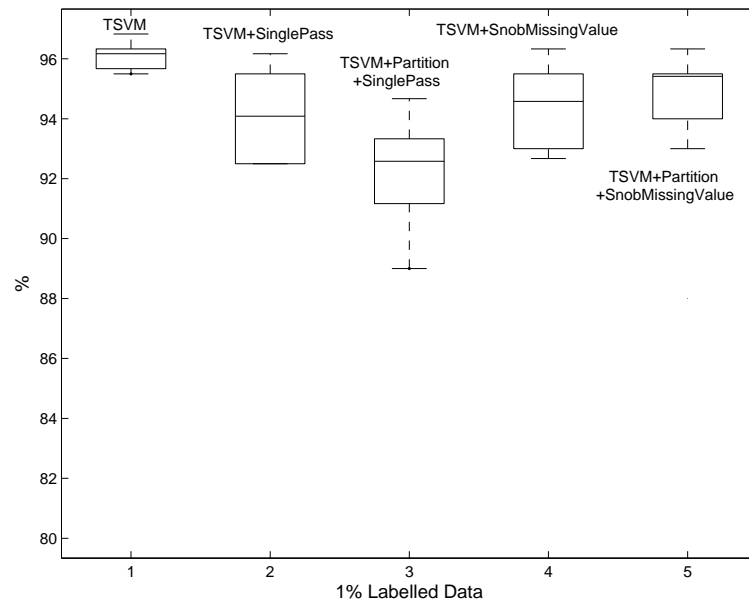


Figure 8: TSVM accuracy scores for 1% labelled data on TF-IDF Reuters

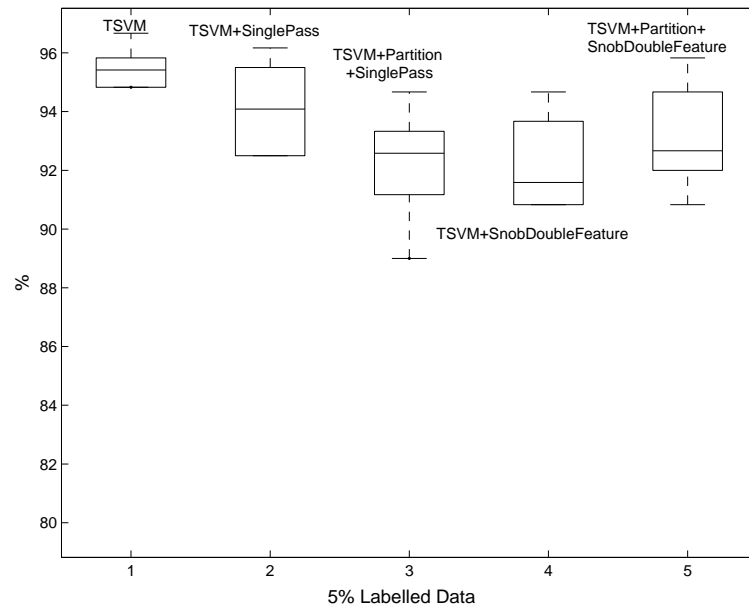


Figure 9: TSVM accuracy scores for 5% labelled data on TF-IDF Reuters

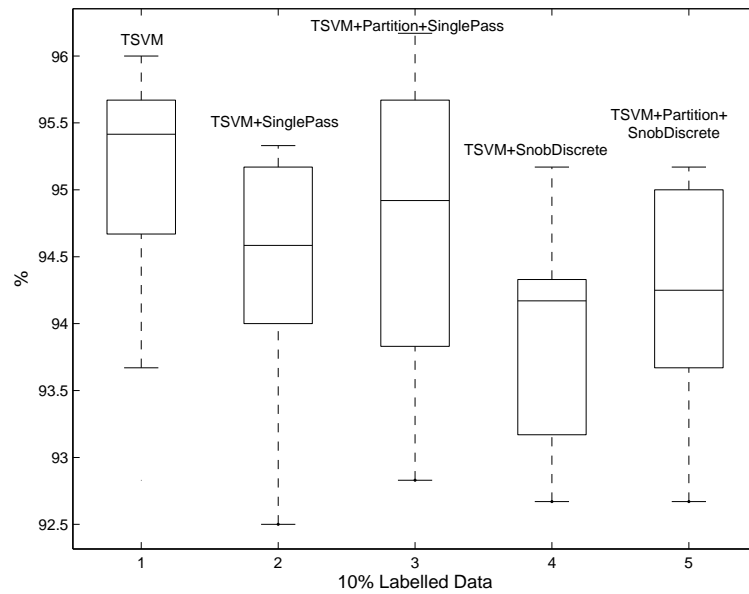


Figure 10: TSVM accuracy scores for 10% labelled data on TF-IDF Reuters

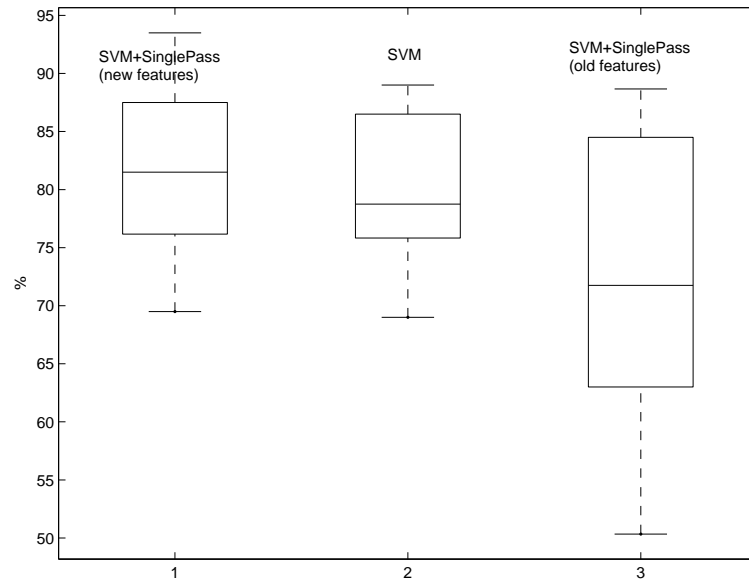


Figure 11: SVM accuracy scores for 1% labelled data on TF-IDF Reuters with new cluster features

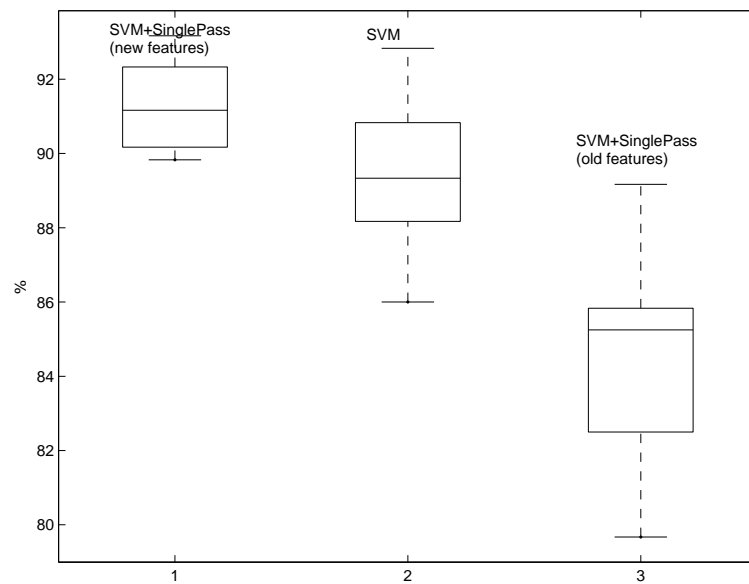


Figure 12: SVM accuracy scores for 5% labelled data on TF-IDF Reuters with new cluster features

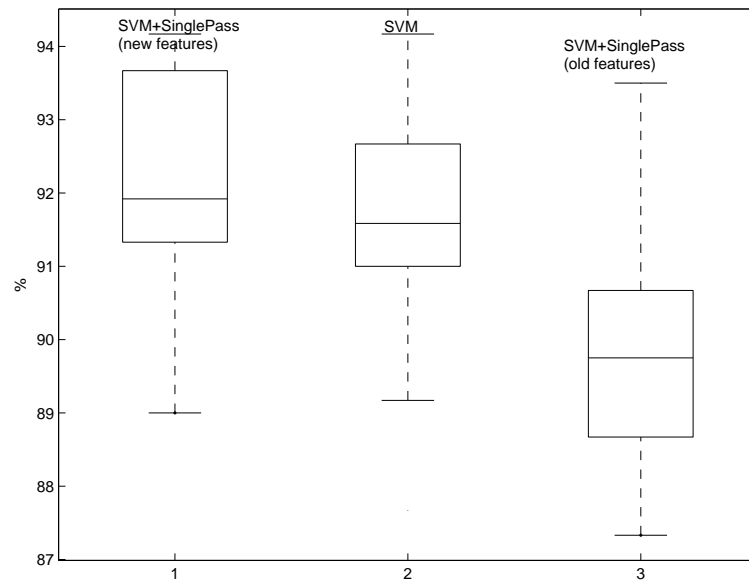


Figure 13: SVM accuracy scores for 10% labelled data on TF-IDF Reuters with new cluster features

Feature Distributions in TF-IDF Reuters

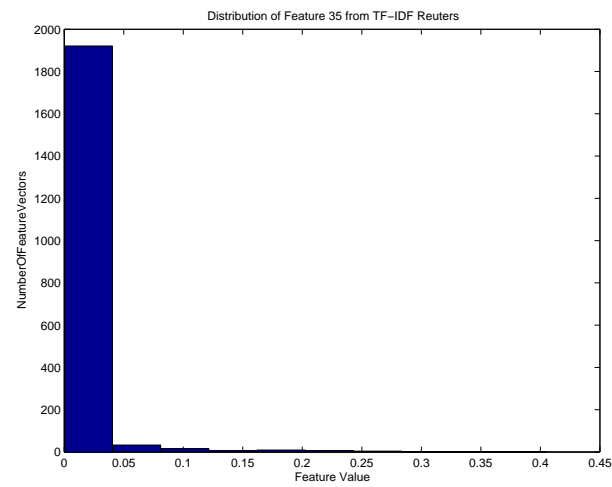


Figure 14: Feature distribution of feature no. 13 from TF-IDF Reuters

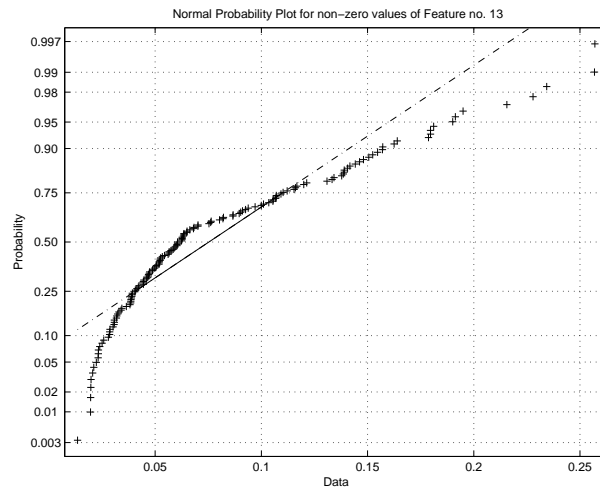


Figure 15: Normal probability plot for non-zero values of feature no. 13 before transformation

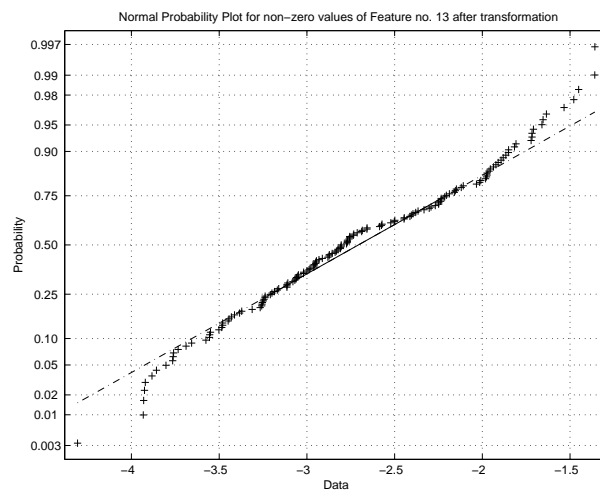


Figure 16: Normal probability plot for non-zero values of feature no. 13 after transformation

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.
- Box, G. E., Hunter, W. G. and Hunter, J. S. (1978). *Statistics for Experimenters*, Wiley, New York.
- 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.
- 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
- 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. (1999a). Making large-scale SVM learning practical, *Advances in Kernel Methods - Support Vector Learning* .

- Joachims, T. (1999b). 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.
- Kohavi, R. and Provost, F. (1998). Special issue on applications and the knowledge discovery process, *Machine Learning* **30**.
- 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. 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.
- Needham, S. and Dowe, D. (2001). Message length as an effective ockham’s razor in decision tree induction, *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics 2001*, Morgan Kaufmann Publishers, CA, Florida, USA, pp. 253–260.
- 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.
- Press, W. H. (1992). *Numerical recipes in C : the art of scientific computing*, Cambridge University Press, Cambridge, New York.

- 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.
- Sebastiani, F. (2002). Machine learning in automated text categorization, *ACM Computing Surveys* **34**: 1–47.
- 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. and Boulton, D. M. (1968). An information measure for classification, *The Computer Journal* **11**: 185–194.
- 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
- 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.