The Impact of Semi-Supervised Clustering on Text Classification

Antonia Kyriakopoulou
Computer Science Department
Athens University of Economics and Business
Patision 76, Athens, 10434, Greece
tonia@aueb.gr

Theodore Kalamboukis
Computer Science Department
Athens University of Economics and Business
Patision 76, Athens, 10434, Greece
tzk@aueb.gr

ABSTRACT

This paper addresses the problem of learning to classify texts by exploiting information derived from clustering both training and testing sets. The incorporation of knowledge resulting from clustering into the feature space representation of the texts is expected to boost the performance of a classifier. Two different approaches to clustering are described, an unsupervised and a semi-supervised one. We present an empirical study of the proposed algorithms on a variety of datasets. The results are encouraging, revealing that information resulting from clustering can create text classifiers of high-accuracy.

Categories and Subject Descriptors

I.5 [**Pattern Recognision**]: Clustering; I.2 [**Artificial Intelligence**]: Learning—*Induction*

General Terms

Experimentation, Performance, Algorithms

Keywords

Text classification; clustering; semi-supervised clustering.

1. INTRODUCTION

Text classification and clustering are typical tasks of text mining[7]. In traditional supervised classification, an inductive learner is first trained on a training set and then is called to classify a testing set, about which it has no prior knowledge. An ideal case would be for the classifier to have information about the distribution of the testing examples before it classifies them. In the unsupervised case, the aim is to extract a kind of "structure" from a given sample of objects and learn a concise representation of these data. The reasoning behind this is that if some structure exists in the objects, it is possible to take advantage of this information and find a short description of the data. In [13] an algorithm (referred to as *baseline* algorithm) that combines supervised and unsupervised classification is proposed. In the baseline algorithm, given a classification problem, the training and testing examples are both clustered before the classification step in order to extract the "structure" of the whole dataset, exploiting the dependence or association between index terms and documents. The structure extracted from the dataset is "translated" in such a way that each cluster is represented by one representative. This concise representation of the whole dataset is incorporated in the existing data representation; each object is assigned the corresponding cluster id using appropriate artificial *meta*-features. After extensive experimentation and analysis of the baseline algorithm it was found that the use of prior knowledge about the nature of the testing set derived from clustering significantly helped in building a more efficient classifier for this set.

In this paper, we present a refined version of the baseline algorithm described in [13]. After going through a rethinking of its basic steps, two main issues emerged and triggered the procedure of refinement. The first issue deals with the nature and generation of the datasets and classification tasks and the second deals with the integration of knowledge from the classification task into the clustering task. Corrective actions have been taken, leading to a re-organizing of the clustering step of the baseline algorithm. In this case, a semi-supervised clustering algorithm is proposed, applied, and tested with encouraning results.

The paper is organized as follows. Section 2 presents the research in the field. Section 3 briefly describes the baseline algorithm. Sections 4 and 5 describe the refined baseline algorithm. In Section 6, the experimental settings and results are presented. Finally, Section 7 concludes with a summary of the work.

2. RELATED RESEARCH

Text classification and clustering have been the focus of critical research in the areas of machine learning and artificial intelligence. In the literature, these two streams flow independently of each other, despite their akin close conceptual and practical connections. However, there are several important research issues encapsulated into text classification tasks and the role of clustering in support of these tasks is also of great significance.

A standard research issue for text classification is the creation of compact representations of the feature space and the discovery of the complex relationships that exist between features, documents and classes. There are several approaches that try to quantify the notion of information

PCI 2013 September 19 - 21 2013, Thessaloniki, Greece

http://dx.doi.org/10.1145/2491845.2491866

for the basic components of a text classification problem. Given the variables of interest, sources of information about these variables can be compressed while preserving their information. Clustering is one of the approaches used in this context. In this vein, an important area of research where clustering is used to aid text classification is the area of dimensionality reduction. Clustering is used as a *feature compression and/or extraction method*: features are clustered into groups based on selected clustering criteria and new, reduced-size event spaces are created by joining similar features into groups. Two types of clustering have been identified: i) one-way clustering, i.e. feature clustering based on the distributions of features in the documents or classes[2, 17], and ii) co-clustering, i.e. clustering both features and documents[5].

Clustering in large-scale classification problems is another major research area in text classification. A considerable amount of work is done on using clustering to reduce the training time of a classifier when dealing with large data sets. In particular, while SVM classifiers[21] have been proved to be a great success in many areas, their training time is at least $O(N^2)$ for training data of size N, which makes them non-favorable for large data sets. The same problem applies to other classifiers as well. In this vein, clustering is used as a down-sampling pre-process to classification in order to reduce the size of the training set resulting in a reduced dimensionality and a smaller, less complex classification problem, easier and quicker to solve. However, it should be noted that dimensionality reduction is not accomplished directly using clustering as a feature reduction technique as discussed earlier, but rather in an indirect way through the removal of training examples that are most probably not useful to the classification task and the selection of the most representative redundant training set. In most of the cases[24, 1], this involves the collaboration of both clustering and classification techniques.

A third research area of text classification where clustering has a lot of impact is the area of *semi-supervised learning*. Training data contain both labeled and unlabeled examples. Obtaining a fully labeled training set is a difficult task; labeling is usually done using human expertise, which is expensive, time consuming and error prone. Obtaining unlabeled data is much easier since it involves collecting data that are known to belong to one of the classes without having to label them. Clustering is used as a method to extract information from the unlabeled data in order to boost the classification task. In particular, clustering is used: i) to create a training set from the unlabeled data[6], ii) to augment the training set with new documents from the unlabeled data[25, 26], iii) to augment the dataset with new features[19, 12, 15], and iv) to co-train a classifier[14, 10].

The work presented in this paper falls into the third category of the semi-supervised research area where clustering is used to augment the dataset with new features. There exist two similar efforts[19, 15] presented in detail next. First, in [19], given the co-occurrences of features and documents of the training set, the features are first hard clustered. Let H be the reduced matrix resulting from clustering. The relation between a feature vector $\vec{d}$ and its reduced vector $\vec{s}$ is $H\vec{d} = \vec{s}$. Next, the two vectors are concatenated into a vector $\vec{d'}$. Then, the testing set is classified with SVM using $\vec{d'}$ as input. The expansion of the feature space is equivalent to using a special kernel in the original feature space, where the form of the mapping to a higher dimensional space depends on the given data. Experiments conducted on several datasets show that the method is effective especially when the training set is small.

Also, in [15], the feature space is augmented with new features derived from clustering the labeled and unlabeled data[1]. A non-hierarchical single-pass clustering algorithm is used to cluster labeled and unlabeled examples. The testing examples are not clustered. In order to derive only the useful information from the clusters, the clusters are sorted by their sizes, and the largest N clusters are chosen as representatives of the major concepts. Each cluster contributes the following features to the feature space of the labeled and the testing examples: i) a binary feature indicating if this is the closest of the N clusters, ii) similarity of the example to the cluster's centroid, iii) similarity of the example to the cluster's unlabeled centroid, i.e. the average of the unlabeled examples that belong to the cluster, and iv) for each class in the labeled set, similarity of the example to the cluster's class l-centroid defined as the average of the examples in class l that belong to this cluster. The clusters are thought of as higher level "concepts" in the feature space and the features derived from the clusters indicate the similarity of each document to these concepts. The unlabeled data are used to improve the representation of these concepts, i.e. to improve the training set. They evaluate the method using SVM classifiers on well-known corpora and find significant improvements in the classification performance.

The work in [15] can be considered as the more relevant to the work of the authors presented here. The difference of the algorithm presented in this paper is that its goal is to introduce a kind of bias into the learner by "highlighting" the dependences between the examples of the training and testing sets before classification. It is assumed that the dependences can be captured by clustering both training and testing examples as a former step to classification. No unlabeled examples are used. Clustering can provide information about the joint probability distribution over words of the training and testing data other than the class label. Then, using the clustering results (i.e. the cluster id of each example) corresponding artificial features are assigned to each feature vector. We know that this process is problematic due to the complexity of texts and the unsupervised nature of clustering. Nevertheless, these assumptions encode a relationship between the training and testing examples that allows for increasing classification rates, specifically when training data are scarce.

For a detailed review and interpretation of the role of clustering in different fields of text classification see [11].

3. THE BASELINE ALGORITHM: USING CLUSTERING TO IMPROVE CLASSIFICATION

Consider a k-class categorization problem, $(k \geq 2)$, with a labeled training sample $Tr = \{(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)\}$ of feature vectors $\vec{x} \in \mathcal{R}^n$ and corresponding labels $y_i \in \{1, \ldots, k\}$, and an unlabeled testing sample $Te = \left\{\vec{x_1^*}, \ldots, \vec{x_m^*}\right\}$ of fea-

[1]A strict distinction should be made between the unlabeled and the testing examples of a dataset. The unlabeled examples are part of the training set. The testing examples are the ones that are to be classified.

ture vectors. We are mostly interested in the case where $Te \gg Tr$. The features are valued using the *TF*IDF* weighting scheme[16], defined by

$$W(f_i) = TF(f_i, \vec{x}) * IDF(f_i) \quad (1)$$

where the *term frequency* of feature f_i, $TF(f_i, \vec{x})$, is the number of its occurrences in document $\vec{x}$, the *inverse document frequency* is

$$IDF(f_i) = \log_2 \left(\frac{|D|}{DF(f_i)} \right) \quad (2)$$

where $|D| = |Tr \cup Te|$ is the number of documents in the dataset and the *document frequency*, $DF(f_i)$, is the number of documents that contain f_i at least once. All feature vectors are normalized to unit length.

3.1 The clustering step

For the classification task, we assume that classes correspond to thematic topics. This assumption corresponds to an ideal case of clustering where all the examples of a class will be clustered together since they share the same word distribution. Therefore, we have considered that there is a one-to-one correspondence between classes, topics and clusters. Hence, in the clustering step of the algorithm, the number of clusters is chosen to be equal to k, i.e. the predefined number of classes. At present, the experimental results verify our conjecture although other clustering algorithms should also be tested in this step.

The CLUTO Clustering Toolkit[9] is used and a divisive clustering algorithm with repeated bisections is selected for clustering both the training and testing sets. In this method, the desired k-way clustering solution is computed by performing a sequence of $k-1$ repeated bisections. The dataset is first clustered into two groups and then one of these groups is selected and disected further. This process continues until the desired k number of clusters is found. During each step, the cluster is bisected so that the resulting 2-way clustering solution optimizes the internal criterion function

$$\max \sum_{g=1}^{k} \sqrt{\sum_{u,v \in S_g} sim(u,v)} \quad (3)$$

where S_g is the set of documents assigned to the g^{th} cluster, u and v represent two documents, and $sim(u, v)$ is the similarity between two documents. The generated set of clusters $G = \{G_1, G_2, \ldots, G_k\}$ consists of k non-overlapping clusters.

3.2 The expansion step

In the expansion step, each cluster in G contributes one *meta*-feature to the feature space of the training and testing sets, i.e. k *meta*-features are created. The weight of these *meta*-features is computed by applying the *TF*IDF* weighting scheme to the clusters. We assume that all the documents in a cluster G_j share the same *meta*-feature mf_j whose *frequency* within a document $\vec{x}$ of the cluster equals to one, $TF(mf_j, \vec{x}) = 1$, its *document frequency* equals to the size of the cluster, $DF(mf_j) = |G_j|$, and its *inverse document frequency* is $IDF(mf_j) = \log_2 \left(\frac{|D|}{|G_j|} \right)$. Then by properly adjusting Eq. 1 the weight of mf_j is defined by

$$W(mf_j) = \log_2 \left(\frac{|D|}{|G_j|} \right) \quad (4)$$

3.3 The classification step

Finally, in the classification step, the SVM$^{\text{light}}$ implementation[2] of SVMs is used. Following the one-against-all approach, a separate machine is trained for each class using the *expanded* dataset, a linear kernel is used in all cases, and the weight C of the slack variables is set to default.

4. CLUSTERING REVISITED

4.1 Loosening the assumptions of the baseline algorithm

There were three basic assumptions made on the baseline algorithm as described in the previous Section. These assumptions helped into defining and reasoning the clustering step of the algorithm. The first assumption suggested that in the clustering step of the algorithm, all the examples of a class will be clustered together since they share the same word distribution. So we had considered that there is a one-to-one correspondence between classes and clusters. Accordingly, the second assumption made, anticipated that the number of clusters is chosen to be equal to the number of classes. This implies that the number of clusters is predefined. Lastly, although not explicitly mentioned, the third assumption prescribed that no a priori knowledge about the classes is embedded in the clustering step, i.e. the clustering algorithm used is fully unsupervised.

These assumptions imposed strict limitations to the baseline algorithm and in particular to the clustering step. In what follows, we loosen the assumptions and describe how this action interacts with and influences clustering.

Multiple clusters per class.

The assumptions of the basic model framework about the generation of text documents (one-to-one correspondence between clusters and classes, and word independence) are violated in real-world text data. Documents of a (broad) class are often mixtures of multiple topics. Words within a document are not independent of each other–grammar and topicality make this so. Documents in the same category are assumed to be generated from an identical distribution, however, categories are manually defined and there is no predefined probabilistic structure behind them. Faced with data that do not fit the assumptions of our model we extend the basic model so that it more naturally fits the data. We directly approach the problem by removing or weakening these restrictive assumptions of the basic model framework.

Specifically, we relax the assumption of a one-to-one correspondence between classes and clusters and replace it with a less restrictive assumption: a one-to-many correspondence between classes and clusters. For textual data, this corresponds to saying that a class may be comprised of several different sub-topics, each best captured with a different word distribution. Furthermore, grouping the documents of a class with multiple clusters can capture some dependencies between words.

For example, consider a sports class consisting of documents about both tennis and basketball. In these documents, the words "racquet" and "set" are likely to co-occur, and the words "foul" and "jumpball" are likely to co-occur. However, these dependencies cannot be captured by a single

[2] `http://svmlight.joachims.org/`

distribution over words in the sports class, i.e. they cannot be grouped into one cluster. On the other hand, with multiple clusters per class, one cluster can cover the tennis sub-topic, and another the basketball sub-topic; thus more accurately capturing the co-occurrence patterns of the above four words. Another example is the task of spam mail filtering. The goal in this task is to make a personalized spam mail filter for a single user's inbox that correctly classifies its emails as *spam* or *non-spam*. The training sets in such cases are typically derived from multiple Internet sources reflecting different distributions of *spam* and *non-spam* mails that are different from that of the individual user's mails. Consequently, categorization performance should be improved if we cluster texts in order to make the mixture distributions assumption for each class more likely to be true.

Dynamic, auto-adjustable number of clusters.

In this vein, finding the "right" number of clusters k for a class (and dataset) is a difficult and often ill-posed problem, since there can be several answers depending on the scale or granularity one is interested in. In probabilistic approaches to clustering, likelihood-ratios, Bayesian techniques and cross-validation are popular. In non-probabilistic methods, a regularization approach, which penalizes for a large k, is often adopted. If the data is labelled, then mutual information between cluster and class labels can be used to determine the number of clusters. Other metrics such as purity of clusters or entropy are of less use as they are biased towards a larger number of clusters. Otherwise, one can employ a suitable heuristic to obtain an appropriate value of k during the clustering process[18].

Semi-supervised clustering.

In semi-supervised clustering it is not only the inherent structure of the data that drives cluster formation, but also the description language which is available to the learner. Provided with knowledge of the given category structure (i.e., the class labels of the training examples), semi-supervised clustering assumes that the examples are classified. Unlike traditional clustering, the goal of semi-supervised clustering is to identify class-uniform clusters that have high probability densities with respect to a single class. New examples are assigned to clusters using a notion of closeness with respect to a given distance function. According to [20] 'semi-supervised clustering employs limited supervision in the form of labelled instances or pairwise instance constraints to aid unsupervised clustering and often significantly improves the clustering performance'.

Existing methods for semi-supervised clustering can be generally grouped into three categories. First, the *constraint-based* methods aim to guide the clustering process with pairwise instance constraints [22] or initialize cluster centroids by labelled instances [3]. Constraints typically specify that two examples must be in the same class (must-link) or must be in different classes (cannot-link). Second, the *distance-based* methods employ metric learning techniques to get an adaptive distance measure used in the clustering process based on the given pairwise instance constraints [23]. Finally, the *hybrid method* proposed by Basu et al. [4] unifies the first two methods under a general probabilistic framework.

The specifications for a new clustering algorithm.

By loosening the assumptions of the baseline algorithm and with the above corrective actions in mind, we constructed the specifications for a new clustering algorithm that aims at better capturing information about the training and testing examples. Provided with knowledge of the given category structure (i.e., the class labels of the training examples), the proposed clustering algorithm will improve the quality of clusters. It will allow for the creation of multiple clusters per class so that it better represents the possible many different word distributions included in the class, and lastly, in order to manage this it will be able to discover the different distributions and adjust the number of clusters created accordingly. The algorithm follows a kind of constraint-based method to guide the clustering process by not allowing training examples that belong to different categories to be grouped together in the same cluster.

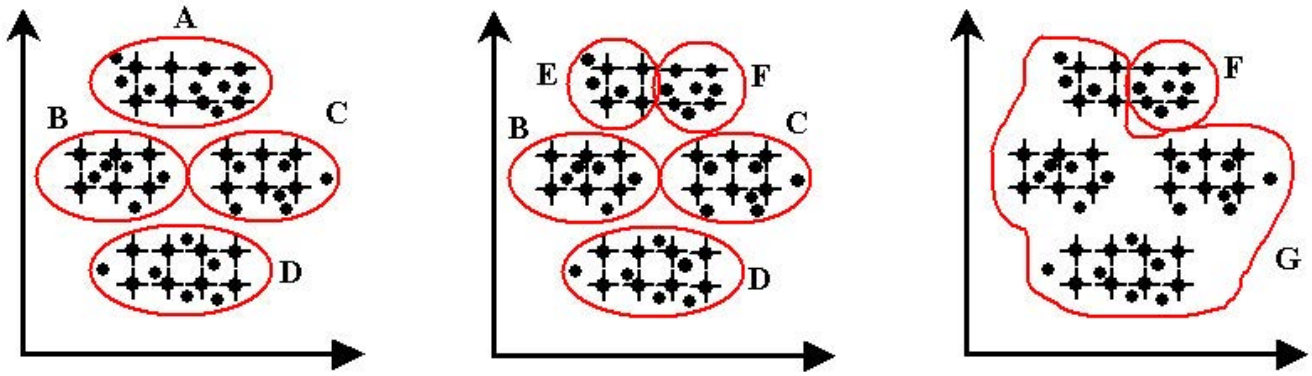
In the next Sections, a new clustering algorithm called *backward semi-supervised clustering (BS^{2}C)* is proposed in order to replace the clustering algorithm used in the clustering step of the baseline algorithm. Its results are incorporated into the feature space representation of the documents in the same way as in the baseline algorithm. It is expected that the prior knowledge about the nature of the testing examples *and* of the categories structure will support the construction of a classifier that will be more efficient than that of the baseline algorithm. Indeed, experiments conducted on several datasets demonstrate the effectiveness of the proposed clustering algorithm when combined with a classifier, against standard classifiers and the baseline algorithm, especially for small training sets.

5. BACKWARD SEMI-SUPERVISED CLUSTERING (BS^{2}C)

Assume that the dataset consists of training and testing examples that belong to k classes. The algorithm follows a one-against-all approach, each time having information about the positive and negative examples of one class. First, a common (unsupervised) clustering algorithm–in our case the well-known k-means algorithm–is applied to the whole dataset, both training and testing examples, and divides the dataset into two clusters. These two clusters will contain a portion of the testing examples, but their conciseness of positive and negative examples is of the essence, and constitutes the next step of the algorithm. Essentially, if the resulting clusters contain both positive and negative examples for the class (apart from the testing examples), then they are selected for further bisection by k-means. This process of continuous cluster bisections is repeated until the clusters created are homogeneous, i.e. they contain only positive or only negative examples for the class, as well as the portion of the testing examples. In some cases, a cluster may contain only testing examples and no training examples. These clusters are not bisected any further, but instead they are all unified in one cluster, G^T. When the procedure of repeated bisections finishes, all the clusters with the positive examples are merged into one large cluster, G^+, and the same is applied for the clusters that contain the negative examples, they are all merged in a large cluster G^-. The examples in cluster G^T need to be integrated in the G^+ or G^-. This is done with the following procedure. For each set of examples belonging to clusters G^+ and G^-, we compute their centroid

Figure 1: Differences between Traditional Clustering and Backward Semi-Supervised Clustering.

(a) Traditional clustering (b) Bisecting promiscuous clusters (c) BS^{2}C result



vectors. We define the *mean* or *centroid vector* μ_h to be

$$\mu_h = \frac{1}{|G^h|} \sum_{x \in G^h} \vec{x}, \text{ for } h = \{+, -\} \quad (5)$$

which is nothing more than the vector obtained by averaging the weights of the various terms present in the examples x of G^+ and G^- respectively. This leads to the creation of 2 centroid vectors $\{\mu_+, \mu_-\}$. The cluster of an example $x \in G^T$ is determined as follows. First, we compute the similarity between $\vec{x}$ to the 2 centroids using the cosine measure. Finally, based on these similarities, we assign x to the cluster corresponding to the most similar centroid. That is, the cluster of x is given by

$$arg \min_h ||x - \mu_h||^2 \quad (6)$$

In this setting, given the sets of the positive, Tr^+, and negative, Tr^-, training examples for a class, the set Te of the testing examples, and the corresponding clusters G^+, G^-, G^T, Table 1 describes a pseudo-code for the *backward semi-supervised clustering* algorithm in our approach.

What we have accomplished is to inject a kind of *backward supervision* to the unsupervised clustering algorithm by examining the result of each iteration. Figure 1 illustrates the differences between traditional and *backward semi-supervised* clustering.

Let us assume that the + and − signs in the Figure represent the positive and negative examples of a class, while the dots represent the testing examples. A traditional clustering algorithm would, very likely, identify the four clusters depicted in Figure 1(a). The reason is that traditional clustering is ignorant with respect to class membership of examples. If our objective is to generate a summary for the class, the clustering in Figure 1(a) would not be very attractive since it combined positive and negative examples in cluster A and put positive examples of the class in three different clusters B, C and D. Backward supervised clustering algorithm on the other hand, maximizes class purity and would split cluster A into two clusters E and F as seen in Figure 1(b). Another characteristic of backward semi-supervised clustering is that it keeps the number of clusters low. Consequently, clusters E, B, C and D would be merged into one cluster without compromising class purity while reducing the number of clusters. Backward semi-supervised clustering algorithm would identify cluster G as the union of clusters E, B, C and D as illustrated by Figure 1(c).

Table 1: The *backward semi-supervised clustering* algorithm.

Foreach class let:
$D = (Tr^+ \cup Tr^-) \cup Te$
$G = G^+ = G^- = G^T = \emptyset$
cluster $(D, 2, G)$
repeat
foreach $(G_j \in G), j = \{1, 2\}$
if $(G_j \subset (Tr^+ \cup Te))$ and $(G_j \subset (Tr^- \cup Te))$
cluster $(D_{G_j}, 2, G)$
if $(G_j \subset (Tr^+ \cup Te))$
merge (D_{G_j}, G^+)
if $(G_j \subset (Tr^- \cup Te))$
merge (D_{G_j}, G^-)
if $(G_j \subset Te)$
merge (D_{G_j}, G^T)
$G = G - G_j$
end
until $G = \emptyset$
$\mu_h \leftarrow \frac{1}{|G^h|} \sum_{x \in G^h} x$, for $h = \{+, -\}$
foreach $x \in G^T$
assign each data point x to the cluster G^*,
for $G^* = arg \min_h ||x - \mu_h||^2$
end

6. EMPIRICAL ANALYSIS

6.1 Experiment settings

The empirical evaluation was done on three *single*-labelled datasets: the "by-date" version of 20Newsgroups[3], WebKB[4], and a subset of Reuters21578[5], Reuters8. This subset origi-

[3] `http://qwone.com/ jason/20Newsgroups/`.
[4] `http://www.cs.cmu.edu/~WebKB/`.
[5] `http://www.daviddlewis.com/resources/testcollections/`.

nated from Reuters10 subset, when considering only the documents with a single topic and the classes which still have at least one train and one test example. We also tested the algorithm on two spam mail filtering tasks (Tasks A and B) provided at the ECML-PKDD 2006 Discovery Challenge[6]. The goal in both tasks was to make a personalized spam filter for a single user's inbox that correctly classified its emails as *spam* or *non-spam*. The reason we have choosen to test the algorithm in these tasks was because we believed that they vitiate in the most straightforward way the assumptions of the baseline algorithm, and for this reason they constitute a great challenge for its refinement, the BS^{2}C-SVM.

Indeed, the majority of the supervised machine learning techniques presented for spam mail filtering assume that emails are drawn independently from a given distribution. That is, the statistical distribution of emails in the training dataset is identical to that of the individual user's emails on which the trained filter will be applied. This assumption, however, is usually incorrect in practice; the training dataset is typically derived from multiple Internet sources reflecting different distributions of spam and non-spam emails that are different from that of the individual user's emails. We believe that when the training and testing sets of one class consist of examples that follow different word distributions like in the case of spam filtering, BS^{2}C will manage to capture this promiscuous information and integrate it into one common parameter, a cluster. Also, an advantage of BS^{2}C is that it does not need to know the number of topics-clusters beforehand. Since the number of topics in a broad class such as spam mails is usually unknown we can not make safe assumptions for the number of clusters the class contains. The task specific number of emails and inboxes can be found in Table 2.

Table 2: The datasets used in the spam mail filtering tasks.

	Task A	**Task B**
Number of labelled training emails	4000	100
Number of emails within one evaluation inbox	2500	400
Number of inboxes for evaluation	3	15

For WebKB the settings in [8] were followed. For the rest of the corpora no feature selection was done, both stemming and stopword-removal were used. Where applicable a four-fold cross validation is done. The algorithm runs four rounds with different samples from the training set. These samples are selected randomly and uniformly by dividing the training set in smaller and equal parts and by preserving the same proportion of documents per category with that of the original dataset. All testing documents are used. For comparison reasons the precision/recall breakeven point (BEP) and the area under curve (AUC) were used as measures of performance per case.

6.2 Results and analysis

Several experiments were conducted using different sizes of the training set. To provide a baseline for comparison, results from the standard SVM classifier, the baseline algorithm (C-SVM), and the refined baseline algorithm (BS^{2}C-SVM) that uses BS^{2}C algorithm in the clustering step, are presented. Tables 3 to 5 show the results of the experiments. The bold entries correspond to the best classification performance for each experiment.

In the case of the spam mail filtering tasks, we can see that in Task B (see Table 3) the baseline algorithm does not manage to capture the intricacies of the text. Its performance is almost the same with that of the standard SVM, with a few exceptions where it performs slightly better. On the other hand, the BS^{2}C-SVM algorithm outperforms both classifiers in all cases, increasing the AUC over 29% in some users' inboxes. Clearly, BS^{2}C clustering algorithm manages to capture the promiscuous information and reflect the different distributions of spam and non-spam mails.

Table 3: AUC for Task B users.

Spam Mail Filtering - Task B			
Users	SVM	C-SVM	BS^{2}C-SVM
user00	63.92	67.00	**93.69**
user01	64.89	66.84	**87.64**
user02	82.06	82.73	**85.66**
user03	91.44	92.92	**92.00**
user04	85.59	86.22	**94.33**
user05	86.74	87.88	**94.40**
user06	75.40	75.55	**76.99**
user07	79.00	79.42	**87.30**
user08	79.88	81.59	**95.13**
user09	71.20	73.14	**79.65**
user10	88.44	89.17	**90.73**
user11	88.31	88.93	**88.50**
user12	77.56	78.83	**95.09**
user13	82.92	85.81	**86.40**
user14	76.16	77.04	**79.59**
Average AUC	**79.56**	**80.87**	**88.47**

For Task A, since the size of the dataset was large (compared to that of Task B), we explored the effect of our approaches on different sizes of the training set. As seen in Table 4, the baseline algorithm has a similar perfomance to the standard SVM as in Task B. When the size of the training set is small BS^{2}C-SVM again outperforms both the other two classifiers confirming the results from the experiments on the single-labelled datasets. However, when the training set increases it does not manage to keep its competitive advantage. Our conjecture is that when the training set is large, the BS^{2}C clustering algorithm suffers from an "overfitting" of the training set. By looking at the intermediate results of the runs of the algorithm we show that the clusters constructed at each iteration of the algorithm were small and most of them contained only training examples. As a result, when these sub-clusters were merged (into the clusters with the positive and negative examples of the dataset) as prescribed in the algorithm, only a small percentage of the testing examples had been "identified" as positive or negative for the spam class from the algorithm and had been included in these clusters. This means that the algorithm failed to correlate the training and testing examples, and this was transferred to the classification task.

In the case of the single-labelled datasets (see Table 5), BS^{2}C-SVM algorithm outperforms both standard SVM and the baseline algorithm when the size of the training set is very small. When the size of the training set increases BS^{2}C-SVM still performs better than standard SVM, however the

[6]Available at `http://www.ecmlpkdd2006.org/challenge.html`.

Table 4: Average AUC for different sizes of the training set for Task A users.

Spam Mail Filtering - Task A									
% train	user00			user01			user02		
	SVM	C-SVM	BS^{2}C-SVM	SVM	C-SVM	BS^{2}C-SVM	SVM	C-SVM	BS^{2}C-SVM
0.5	73.86	73.99	**85.16**	84.56	85.83	**91.33**	73.86	74.57	**79.57**
1	71.19	71.16	**86.49**	81.33	82.35	**85.08**	76.60	77.11	**87.86**
5	72.74	72.70	**89.18**	79.42	79.89	**83.54**	88.96	89.69	**90.46**
10	78.26	78.24	**79.48**	85.71	**86.30**	81.43	88.96	89.58	**96.84**
20	**80.20**	80.12	77.38	85.83	**86.01**	82.19	90.79	**90.82**	89.90
50	**83.06**	83.01	82.28	**88.26**	87.96	87.79	**93.29**	93.27	92.21
100	**85.25**	84.89	81.42	**89.92**	89.73	88.65	94.67	94.42	**94.86**

Table 5: Average BEP for different sizes of the training sets for the single-labelled datasets.

% train	20 Newsgroups			WebKB			Reuters8		
	SVM	C-SVM	BS^{2}C-SVM	SVM	C-SVM	BS^{2}C-SVM	SVM	C-SVM	BS^{2}C-SVM
0.5	40.39	56.51	**61.99**	48.32	58.38	**60.15**	54.70	56.50	**62.43**
1	50.26	63.47	**66.31**	49.28	56.97	**65.58**	69.83	71.04	**74.77**
5	67.98	**72.19**	70.95	51.37	58.66	**67.18**	80.67	**81.33**	79.08
10	72.29	**75.14**	73.79	56.59	60.45	**67.26**	85.42	**86.72**	82.62
20	76.34	**77.79**	76.71	64.14	**68.95**	67.49	88.42	**89.56**	89.40
50	80.29	**81.04**	80.26	69.08	**72.65**	69.20	90.17	**90.84**	90.68
100	82.35	82.79	**83.00**	72.85	**74.28**	73.52	91.81	**93.23**	91.35

baseline algorithm performs slightly better.

7. CONCLUSIONS

Some datasets cannot be adequately modelled with the basic clustering process of the baseline algorithm. In these cases, using clustering to capture information about the feature distribution can actually decrease classification performance by finding clusters that only poorly match the true data distribution. When a class is complex, with several or many sub-topics, a single cluster as a representative of the class is insuficient; a different approach is required. In this paper, we used class-knowledge and a semi-supervised clustering algorithm on a spam filtering task (amongst other) to describe the single spam class containing emails about many topics from different distributions. Using the hard, unsupervised clustering algorithm of the baseline algorithm here had little positive impact to classification performance. However, the semi-supervised clustering algorithm with many clusters per class was representative enough for this dataset to increase classification performance compared to using unsupervised clustering alone.

Also, preliminary results on multi-labelled datasets reveal an inefficiency of clustering to successfully capture the structure of the datasets. Our methods offer little improvement to classification performance when applied on multi-labelled datasets (i.e. datasets where a document may belong to more than one classes) like Reuters21578 and Ohsumed. One promising direction should be to adopt a different, fuzzy approach to clustering instead of the hard clustering approach followed now. In fuzzy clustering, documents can belong to more than one clusters, and associated with each document is a set of membership levels. These indicate the strength of the association between that documents and a particular cluster. Fuzzy clustering is a process of assigning these membership levels, and then using them to assign documents to one or more clusters. This would require a re-construction of the clustering and augmentation steps of the algorithms, but it would be very interesting if they could be applied on these type of datasets.

Further investigation is required for the effect of overfitting that appears when semi-supervised clustering is applied on large training data. While in the domain of supervised classification, practitioners are well aware of the effect of overfitting, it seems like this effect has been completely overlooked in the clustering domain. A better approach in this case, might improve the semi-supervised clustering algorithm even more.

To conclude, in this refinement to the baseline algorithm scheme, we studied the effect of multiple clusters per class and the use of class-knowledge for semi-supervised clustering. This was an effort to relax the assumptions imposed by the basic model, and make the clustering algorithm better match the data. The BS^{2}C algorithm managed to capture information from the training and testing sets, which was successfully incorporated into classification. Its application in cases where the training set for a class consists of examples that follow different word distributions and the number of clusters is not known, is promising and should be further investigated. Experimental results show improvements in classification, and suggest the exploration of even more complex clustering models that would correspond even better to textual data distributions.

8. ACKNOWLEDGMENTS

The work reported in this paper has been funded by the Athens University of Economics and Business under the grant BRFP II.

9. REFERENCES

[1] M. Awad, L. Khan, F. Bastani, and I. L. Yen. An effective support vector machines (SVMs) performance using hierarchical clustering. In *Proceedings. 16th*

IEEE International Conference on Tools-with-Artificial Intelligence. 2004: 663-7, 2004.

[2] L. D. Baker and A. McCallum. Distributional clustering of words for text classification. In *SIGIR*, pages 96–103. ACM, 1998.

[3] S. Basu and A. Banerjee. Semi-supervised clustering by seeding, Aug. 02 2002.

[4] S. Basu, M. Bilenko, and R. J. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 42–49, 2004.

[5] I. S. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, 2003.

[6] G. Fung and O. L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. Technical report, 2001.

[7] M. Hearst. Untangling text data mining. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3–10, 1999.

[8] T. Joachims. Transductive inference for text classification using support vector machines. In *16th International Conference on Machine Learning*, pages 200–209. San Francisco: Morgan Kaufmann, 1999.

[9] G. Karypis. Cluto a clustering toolkit, 2002.

[10] A. Kyriakopoulou. Using clustering and co-training to boost classification performance. In *ICTAI (2)*, pages 325–330. IEEE Computer Society, 2007.

[11] A. Kyriakopoulou. Text classification aided by clustering: a literature review. In P. Fritzsche, editor, *Tools in Artificial Intelligence*, pages 233–252. InTech Education and Publishing, 2008.

[12] A. Kyriakopoulou and T. Kalamboukis. Using clustering to enhance text classification. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 805–806. ACM, 2007.

[13] A. Kyriakopoulou and T. Kalamboukis. Using clustering for text classification. *International Journal on Artificial Intelligence Tools*, 20, 3, 2011.

[14] B. Raskutti, H. L. Ferrá, and A. Kowalczyk. Combining clustering and co-training to enhance text classification using unlabelled data. In *KDD*, pages 620–625. ACM, 2002.

[15] B. Raskutti, H. L. Ferrá, and A. Kowalczyk. Using unlabelled data for text classification through addition of cluster parameters. In C. Sammut and A. G. Hoffmann, editors, *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 514–521. Morgan Kaufmann, 2002.

[16] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[17] N. Slonim and N. Tishby. The power of word clustering for text classification. In *Proceedings of the European Colloquium on IR Research, ECIR 2001*. 2001.

[18] A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc. HiPC, Bangalore, LNCS 1970*, pages 525–536. Springer, New York, 2000.

[19] H. Takamura. Clustering approaches to text categorization, 2003.

[20] W. Tang, H. Xiong, S. Zhong, and J. Wu. Enhancing semi-supervised clustering: a feature projection perspective. In P. Berkhin, R. Caruana, and X. Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 707–716. ACM, 2007.

[21] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[22] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1103–1110. Morgan Kaufmann, San Francisco, CA, 2000.

[23] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, pages 505–512. MIT Press, 2002.

[24] H. Yu, J. Yang, and J. Han. Classifying large data sets using SVMs with hierarchical clusters. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 306–315. ACM, 2003.

[25] H.-J. Zeng, X.-H. Wang, Z. Chen, H. Lu, and W.-Y. Ma. CBC: Clustering based text classification requiring minimal labeled data. In *ICDM*, pages 443–450. IEEE Computer Society, 2003.

[26] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *NIPS*. MIT Press, 2003.