DOI: 10.2298/CSIS120130044Z

Clustering based Two-Stage Text Classification Requiring Minimal Training Data

Xue Zhang[1,2] and Wangxin Xiao[3,4]

[1] Key Laboratory of High Confidence Software Technologies, Ministry of Education, Peking University, Beijing 100871, China
[1]School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
[2]Department of Physics, Shangqiu Normal University, Shangqiu 476000, China
jane_zhang@pku.edu.cn
[3] Department of Computer Science, Jinggangshan University, Ji'an 343009, China
[4]School of Traffic and Transportation Engineering, Changsha University of Science and Technology, Changsha 410114, China
wx.xiao@rioh.cn

Abstract. Clustering has been employed to expand training data in some semi-supervised learning methods. Clustering based methods are based on the assumption that the learned clusters under the guidance of initial training data can somewhat characterize the underlying distribution of the data set. However, our experiments show that whether such assumption holds is based on both the separability of the considered data set and the size of the training data set. It is often violated on data set of bad separability, especially when the initial training data are too few. In this case, clustering based methods would perform worse. In this paper, we propose a clustering based two-stage text classification approach to address the above problem. In the first stage, labeled and unlabeled data are first clustered with the guidance of the labeled data. Then a self-training style clustering strategy is used to iteratively expand the training data under the guidance of an oracle or expert. At the second stage, discriminative classifiers can subsequently be trained with the expanded labeled data set. Unlike other clustering based methods, the proposed clustering strategy can effectively cope with data of bad separability. Furthermore, our proposed framework converts the challenging problem of sparsely labeled text classification into a supervised one, therefore, supervised classification models, e.g. SVM, can be applied, and techniques proposed for supervised learning can be used to further improve the classification accuracy, such as feature selection, sampling methods and data editing or noise filtering. Our experimental results demonstrated the effectiveness of our proposed approach especially when the size of the training data set is very small.

Keywords: text classification, clustering, active semi-supervised clustering, two-stage classification.

1. Introduction

The goal of automatic text classification is to automatically assign documents to a number of predefined categories. It is of great importance due to the ever-expanding amount of text documents available in digital form in many real-world applications, such as web-page classification and recommendation, email processing and filtering. Text classification has once been considered as a supervised learning task, and a large number of supervised learning algorithms have been developed, such as Support Vector Machines (SVM) [1], Naïve Bayes [2], Nearest Neighbor [3], and Neural Networks [4]. A comparative study was given in [5]. SVM has been recognized as one of the most effective text classification methods. Furthermore, a number of techniques suitable for supervised learning have been proposed to improve classification accuracy, such as feature selection, data editing or noise filtering, and sampling methods against bias.

A supervised classification model often needs a very large number of training data to enable the classifier's good generalization. The classification accuracy of traditional supervised text classification algorithms degrades dramatically with the decrease of the number of training data in each class. As we know, manually labeling the training data for a machine learning algorithm is a tedious and time-consuming process, and even unpractical (e.g., online web-page recommendation). Correspondingly, one important challenge for automatic text classification is how to reduce the number of labeled documents that are required for building reliable text classifier. This leads to an active research problem, semi-supervised learning. There have been proposed a number of semi-supervised text classification methods, including Transductive SVM (TSVM) [6], Co-Training [7] and EM [8]. A comprehensive review could be found in [9]. By exploring information contained in unlabeled data, these methods obtain considerable improvement over supervised methods with relatively small size of training data set. However, most of these methods adopt the iterative approach which train an initial classifier based on the distribution of the labeled data. They still face difficulties when the labeled data set is extremely small since they will have a poor starting point and cumulate more errors in iterations when the extremely few labeled data are far apart from corresponding class centers due to the high dimensionality.

To address the problem of sparsely labeled text classification, we present a clustering based two-stage text classification method with both labeled and unlabeled data. Experimental results on several real-world data sets validate the effectiveness of our proposed approach. Our contributions can be summarized as follows.

- —We propose a novel clustering based two-stage classification approach that requires minimal training data to achieve high classification accuracy.
- —In order to improve the accuracy of the self-labeled training data by clustering, we propose an active semi-supervised clustering method to cope with data sets of bad separability.
- —On the basis of clustering, we convert the challenging problem of sparsely labeled text classification into supervised one. Thus supervised

classification models and techniques suitable for text classification can be used to further improve the overall performance.

—We conduct extensive experiments to validate our approach and study related issues.

The rest of this paper is organized as follows. Section 2 reviews several existing methods. Our clustering method is given in Section 3 with some analysis. The detailed algorithm is then presented in Section 4. Experimental results are presented in Section 5. Section 6 concludes this paper.

2. Related Work

Clustering has been applied in many sub-domains of the problem of text classification, including feature compression or extraction [10], semi-supervised learning [11], and clustering in large-scale classification problems [12,13]. The following will review several related work about clustering aiding classification in the area of semi-supervised learning. A comprehensive review for text classification aided by clustering can be found in [14].

Clustering has been used to extract information from unlabelled data in order to boost the classification task. There are roughly four cases of semi-supervised classification aided by clustering. In particularly, clustering is used: (1) to create a training set from the unlabelled set [15], (2) to augment an existing labeled set with new documents from the unlabeled data set [11], (3) to augment the data set with new features [8,16], and (4) to co-train a classifier [17,18]. More recently, simultaneous learning frameworks for clustering and classification have been proposed [19,20].

To make use of unlabeled data, one assumption which is made, explicitly or implicitly, by most of the semi-supervised learning algorithms is the so-called cluster assumption that two points are likely to have the same class label if there is a path connecting them passing through regions of high density only. That is, the decision boundary should lie in regions of low density. Based on the ideas of spectral clustering and random walks, a framework for constructing kernels which implement the cluster assumption was proposed in [21]. Also based on cluster assumption, [22] applied spectral clustering to represent the labeled and unlabeled data. By clustering unlabeled data with labeled data using probabilistic and fuzzy approaches, [23] proposed a framework to improve the performance of base classifier with unlabeled data. In text classification, there are often many low-density areas between positive and negative labeled examples because of the high dimensionality and data sparseness. This situation will be worsened with the decrease of the number of training data in each class.

The most related work is the clustering based text classification (CBC) approach [11]. In CBC, firstly, semi-supervised soft k-means is used to cluster the labeled and unlabeled data into k clusters, where k is set to the number of classes in the classification task. p% most confident unlabeled examples from each cluster (i.e. the ones nearest to the cluster's centroid) are added to the

training data set. Then TSVM is trained on the augmented training data set and unlabeled data set. Similarly, *p*% most confident unlabeled examples from each class (i.e. the ones with the largest margin) are added to the training data set. CBC iterates the step of clustering and the step of classification alternatively until there is no unlabeled data left. In CBC, in order to guarantee the labeling accuracy, the value of *p* should be small enough. That is, after the clustering step in each iteration, the training data set is augmented with very few examples. Therefore, the classifier in the following classification step should have an accepted performance with small size of training data set. This put a strong constraint on the selected classification models. CBC can hardly perform well with supervised classification models, e.g. SVM, which will be demonstrated later.

The success of CBC is based on the assumption that even when some of the data points are wrongly classified, the most confident data points, i.e. the ones with largest margin under classification model and the ones nearest to the centroids under clustering model, are confidently classified or clustered. This assumption guarantees the high accuracy of the self-labeled training data and correspondingly the good performance of the algorithm. We separate this assumption into clustering assumption and classification assumption for convenience. However, our empirical experiments show that the assumptions are often violated on data sets of bad separability. Firstly, clustering assumption can't be hold in this case, at least for the soft-constraint k-means [11]. In fact, each cluster's centroid may locate in: 1) the domain of its corresponding true class, 2) the border of its true class and other classes, 3) the domain of other class. The probability that the last two cases occur increases with the degrading of data separability. In the last two cases (we call them cluster bias), CBC will introduce more noise into the training data set in its clustering step, which might make the classification assumption also be violated since the noise will have a big effect due to the initially very few truly labeled training data. Then the following iterative steps will further cumulate more errors.

In sparsely labeled text classification, the extremely few training data make many techniques which are useful for ameliorating data separability, e.g. feature selection, not effective, because the training data can not characterize the whole data set well. When the size of training data set is extremely small, unsupervised learning gives better performance than supervised and semi-supervised learning algorithms. In this paper, we develop an active semi-supervised clustering based two-stage approach to address the problem of sparsely labeled text classification. Different from CBC, our aim is to convert the problem of sparsely labeled text classification into a supervised one by using clustering. Therefore, supervised classification models, e.g. SVM, can be applied, and techniques proposed for supervised learning can be used to further improve the classification accuracy, such as feature selection, sampling methods and data editing or noise filtering. Furthermore, our proposed active semi-supervised clustering method aims to cope with data sets with any separability. The goal of clustering here is to generate enough training data for supervised learning with high accuracy.

3. Active Semi-supervised Clustering

Using clustering to aid semi-supervised classification, the key point lies in that the clustering results can to some extent characterize the underlying distribution of the whole data set. Only in this case, clustering is helpful to augment training data set or extract useful features to improve the performance of classification. Although clustering methods are more robust to the bias caused by the initial sparsely labeled data, empirical experiences show that the results of clustering might also be biased (e.g. the cluster bias of cases 2) and 3)), sometimes heavily, especially on data sets of bad separability. The soft-constrained k-means in CBC can reduce bias in the labeled examples by basing the constraints (the guidance of the initially labeled data) not on exact examples but on their centroid. But it still cannot cope with the bias well in training data on data sets with bad separability.

Table 1 gives two clustering based algorithms to augment training data. They implement the iterative reinforcement strategy. In each iteration, a clustering method is used to cluster the whole data set with the guidance of labeled training data, and then several examples are selected according to some criteria and labeled with the labels of the centroids they belong to. In SemiCC algorithm, the clustering method is soft-constrained k-means adopted in CBC algorithm. The clustering method (we call it as active soft-constrained k-means) in SemiCCAc algorithm is proposed in order to address the cluster-bias problem.

Table 1. Two Clustering Algorithms: *SemiCC* and *SemiCCAc*

Input:

Labeled data set D_l and unlabeled data set D_u, the number of iterations *maxIter*, *p*

Output:

Augmented labeled data set $D_l^{'}$

In**itialize:** $D_l^{'}=D_l$, $D_u^{'}=D_u$, iter=0

Algorithm *SemiCC*:

While *iter*<*maxIter* and $D_u \neq \Phi$

- *iter*=*iter*+1
- Calculate initial centroids:

  $o_i = \frac{1}{n_i}\sum_{\forall j, t_j=i} x_j, i = 1,...,c, x_j \in D_l^{'}$, and set current centroids $o_i^* = o_i$. n_i is the number of examples in $D_l^{'}$ whose label is *i*.
- The labels of the centroids $t(o_i) = t(o_i^*)$ are equal to labels of the corresponding examples.

- Repeat until cluster result doesn't change any more
  - Assign $t(o_i^*)$ to each $x_i \in D_l + D_u$ that are nearer to o_i^* than to other centroids.
  - Update current centroids:

    $o_i^* = \frac{1}{n_i} \sum_{\forall j, t_j = i} x_j, i = 1,...,c, x_j \in D_l + D_u$, n_i is the number of examples in $D_l + D_u$ whose label is *i*.
  - Calculate the nearest centroids o_j^* for each o_i , if $t(o_i) \neq t(o_i^*)$, exit the loop.
- From each cluster, select *p*% examples $x_i \in D_u^{'}$ which is nearest to o_i^*, add them to $D_l^{'}$, and delete them from $D_u^{'}$.

Algorithm *SemiCCAc*:

While *iter*<*maxIter* and $D_u \neq \Phi$

- *iter*=*iter*+1
- Calculate initial centroids:

  $o_i = \frac{1}{n_i} \sum_{\forall j, t_j = i} x_j, i = 1,...,c, x_j \in D_l^{'}$, and set current centroids $o_i^* = o_i$. n_i is the number of examples in $D_l^{'}$ whose label is *i.*
- The labels of the centroids $t(o_i) = t(o_i^*)$ are equal to labels of the corresponding examples.
- Repeat until cluster result doesn't change any more
  - Assign $t(o_i^*)$ to each $x_i \in D_l + D_u$ that are nearer to o_i^* than to other centroids.
  - Update current centroids:

    $o_i^* = \frac{1}{n_i} \sum_{\forall j, t_j = i} x_j, i = 1,...,c, x_j \in D_l + D_u$, n_i is the number of examples in $D_l + D_u$ whose label is *i*.
  - Calculate the nearest centroids o_j^* for each o_i , if $t(o_i) \neq t(o_i^*)$, exit the loop.
- From each cluster, select *p*% examples $x_i \in D_u^{'}$ which is nearest to o_i^* and sort them with descending order of confidences, $x_1,\ldots,x_m$

- **If** the true label of x_1 and x_m equals to $t(o_i^*)$:
  add the m examples to $D_l^{'}$
  delete them from $D_u^{'}$
  else
  add x_1 and x_m with their true labels to $D_l^{'}$
  delete x_1 and x_m from $D_u^{'}$
  end

SemiCCAc is different from SemiCC in the labeling strategy for the selected examples of highest confidences according to the clustering results. In soft-constrained k-means, it doesn't take the found centroids' location into consideration. It just labels the selected examples nearest to each centroid. Therefore, it will introduce much noise into the training data set with the presence of cluster bias. In active soft-constrained k-means, it first estimates the location of each centroid. Only for clusters whose centroids locate within their true classes, it labels all the selected examples nearest to the corresponding centroids with the labels of their centroids. For the clusters with the presence of cluster bias, it just labels two examples with their true label for each cluster.

An important problem in active soft-constrained k-means is how to estimate the location of each cluster's centroid. The strategy used here is to inquire the true labels of two examples (the nearest and the farthest examples to the centroid in the selected *p*% examples) by resorting to an oracle or expert for each cluster. If the two examples have the same label with that of their centroid, then all the *p*% selected examples are labeled with the label of the centroid. Otherwise, only the two examples are added to training data set with their true labels. The strategy is based on the intuition that cluster bias is more likely happened when the two examples have different labels with that of their centroid. When the two examples have the same label, but different from their centroid, the cluster's centroid is most likely locate in the domain of other classes. When one of the two examples has the same label with that of the centroid, the cluster's centroid is most likely locate in the border of the true class and other classes. We can filter out much noise by using this strategy. It is also delightful that cluster bias can be rectified in the following iterations in SemiCCAc by estimating the location of clusters' centroids, which property guarantees the high accuracy of self-labeled training data in spite of the poor starting points.

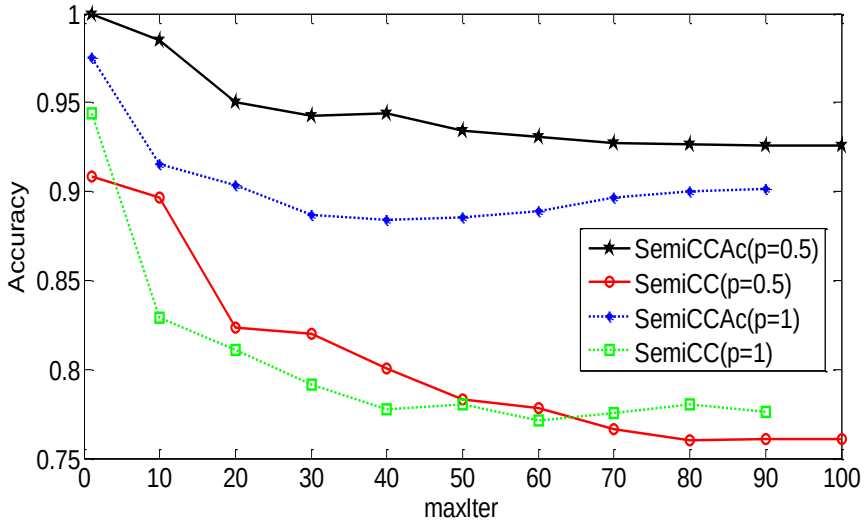

Fig.1. Accuracy of self-labeled training data with iterations

In figure 1, we depict the average accuracy of self-labeled training data by applying the two clustering algorithms to a text classification problem in 20 runs (same2, consisting of two most similar classes in 20Newsgroups, 5 training data for each class). From figure 1, it could be found that the accuracy of self-labeled training data by SemiCCAc is significantly higher than that by SemiCC. With the increase of *p* value, the accuracy degrades first, but then it rises with the increase of iterations.

When *maxIter*=1, *SemiCC* degrades to the soft-constrained *k*-means. We can also see that the average accuracy in SemiCC is below 0.95 when *maxIter*=1, which indicates that soft-constrained *k*-means introduces noise into the training data set with a certain probability. This noise will hurt the following classifier's learning, especially when the size of the initial training data set is very small. We think the phenomenon of cluster bias can partially explain why the performance of CBC improves slowly than those of TSVM and co-training with the increase of training data. With the increase of iterations, the accuracy of self-labeled training data by SemiCC degrades much faster than that of SemiCCAc. Therefore, techniques to cope with the cluster bias are very important for clustering based semi-supervised classification. This also tells us that the proposed active semi-supervised clustering method is effective for addressing the problem of cluster bias.

4. Two-Stage Classification Framework: ACTC

In this section, we present the detail of the Active semi-supervised Clustering based Two-stage text Classification algorithm (ACTC). All documents are tokenized into terms and we construct one component for each distinct term. Thus each document is represented by a vector $(w_{i1}, w_{i2}, ..., w_{ip})$ where w_{ij} is weighted by TFIDF. The cosine function is used in the clustering algorithm to calculate the distance from an example to the centroid. In the classification stage, we use a SVM classifier trained with the augmented training data to classify the whole data set. The detailed algorithm is presented in table 2.

ACTC consists of two stages: clustering stage and classification stage. In the clustering stage, SemiCCAc is used to augment the training data set. Users can set the values of *maxIter* and *p* to determine how many new documents should be labeled by SemiCCAc. At the second stage, discriminative classifiers can subsequently be trained with the expanded labeled data set. Soft-constrained k-means is in fact a generative classifier [11]. According to [24], generative classifiers reach their asymptotic performance faster than discriminative classifiers, but usually lead to higher asymptotic error than discriminative classifiers. This motivates us to combine clustering with discriminative classifiers together to address the problem of sparsely labeled text classification.

ACTC in fact converts the problem of sparsely labeled text classification into a supervised one, thus supervised classification models suitable for text classification can be used. Moreover, the techniques proposed for supervised learning can be used to improve the performance. For instance, it is unavoidable to falsely label some examples in the clustering stage, then data editing or noise filtering techniques are expected to improve the performance of ACTC. Other techniques also can be used to improve the performance, such as feature selection and sampling.

Table 2. ACTC and CBCSVM

Input:

Labeled data set D_l and unlabeled data set D_u and the number of iterations *maxIter*, *p*

Output:

The full labeled set $D_l^{'} = D_l + D_u$

A classifier *L*

Algorithm *ACTC*:

1. **Clustering Stage**
   Use SemiCCAc (repeat *maxIter* iterations) to augment the training data set and we get an augmented training data set $D_l^{'}$
2. **Classification Stage**

Train a SVM classifier *L* based on $D_l^{'}$. And use the learned classifier to classify the whole data set.

Algorithm CBCSVM:

iter=0

1. while *iter*<*maxIter*/2
   iter=*iter*+1
   1.1 Clustering step
   Use soft-constrained k-means to clustering the whole data set, and select *p*% unlabeled examples nearest to its centroid for each cluster and add them to $D_l^{'}$
   1.2 Classification step
   Train a SVM classifier based on $D_l^{'}$.
   From each class, select *p*% unlabeled examples with the largest margin, and add them to $D_l^{'}$
2. Train a SVM classifier *L* based on $D_l^{'}$. And use the learned classifier to classify the whole data set.

In order to verify the two-stage framework performs better than CBC algorithm in supervised learning, we substitute SVM for TSVM in CBC, named CBCSVM. Note that, if we use TSVM as the classifier, the performance of both algorithms will be expected to get improved. However, the time complexity of a TSVM classifier is much higher than that of a SVM classifier, because it repeatedly switches estimated labels of unlabeled data and tries to find the maximal margin hyperplane. The more unlabeled data are, the more time it requires. The worse of the data separability is, the more time it requires. For example, on same2, which consists of two most similar classes of 20Newsgroups and 1000 examples in each class, TSVM requires several hours to complete when 5 training data for each class are used. SVM only needs about 1 second. With enough training data, the performance of SVM is expected to be similar with that of TSVM, but it requires much less time. This motivates us to propose the two-stage classification method, which converts the problem of sparsely labeled text classification into a supervised one.

CBCSVM is also given in table 2 for convenience. Since CBC selects *p*% unlabeled examples both in clustering and classification steps in each iteration, we set the number of iterations to half of that in ACTC in order to make them have the same selection times.

The difference between our approach and CBC is that, we expand the training data set by a self-training style clustering process and resorting to an oracle or expert to evaluate the clusters' centroids. After completion of the training data expansion, discriminative classifiers could be trained on the expanded training data set. Therefore, ACTC puts less constraint on the

classification model, which enables us to treat the following classification stage as a supervised learning problem.

5. Performance Evaluation

5.1. Data sets

For a consistent evaluation, we conduct our empirical experiments on two benchmark data sets, 20NewsGroups and Reuters-21578. 20Newsgroups is one famous Web-related data collection. From the original 20 Newsgroups data set, same2, consisting of 2 very similar newsgroups (comp.windows.x, comp.os.ms-windows) is used to evaluate the performance of the algorithms. Same2 contains 2000 instances, 1000 for each class. We use Rainbow software[1] to preprocess the data (removing stop words and words whose document frequency are less than 3, stemming) and we get 7765 unique terms for same2. Then terms are weighted with their TFIDF values.

The Reuters-21578 corpus contains Reuters news articles from 1987. We only show the experimental results of train1.svm in LWE[2] since the algorithms have the similar performance on other Reuters data sets. Train1.svm contains 1239 documents (two class) and 6889 unique terms.

5.2. Evaluation Metric

We use macro-averaging of F1 measure among all classes to evaluate the classification result.

For each class $i \in [1, c]$, let A_i be the number of documents whose real label is *i*, and B_i the number of documents whose label is predicted to be *i*, and C_i the number of correctly predicted documents in this class. The precision and recall of the class *i* are defined as $P_i = C_i / B_i$ and $R_i = C_i / A_i$ respectively.

For each class, the *F*1 metric is defined as $F_1 = 2 \cdot P \cdot R / (P + R)$ where *P* and *R* are precision and recall for a particular class. *F*1 metric takes into account both precision and recall, thus it is a more comprehensive metric than either precision or recall when separately considered.

The macro-averaging *F*1 is a measurement which evaluates the overall performance of the classification model. It is defined as:

$$Macro\_F1 = \frac{1}{c} \sum_{i=1}^{c} 2 \times P_i \times R_i / (P_i + R_i) \quad (1)$$

[1] http://www.cs.cmu.edu/~mccallum/bow/

[2] http://ews.uiuc.edu/~jinggao3/kdd08transfer

5.3. Experimental Results

The SVM^{light} package[3] is used in our experiments for the implementation of SVM using default configurations.

We first compare ACTC and CBCSVM with different iterations on two data sets. SVM and SemiCCAc are used as the baseline in order to see the benefits brought by our two-stage classification framework and CBCSVM. We set p=0.5. We conduct the experiments 30 runs and the average results are given. The number of training data is 5 for each class and randomly sampled in each run.

Figures 2 and 3 give the Macro_*F*1 performance with different iterations on same2 and Reuters respectively. In ACTC and CBCSVM, parameter *maxIter* determines the number of self-labeled training data. Larger value of *maxIter* means more self-labeled training data and larger size of the training data set for the final SVM training. That is, the size of training data set for the final SVM increases with the increase of *maxIter*.

From figure 2, we can see that ACTC significantly outperforms the other algorithms with any value of *maxIter* and its performance improves with the increase of the *maxIter*. This indicates the following two aspects. One is that SVM classifier significantly benefits from the augmented training data set by comparing its performance with that of SVM trained on the initial training data set. The other is that the self-labeled training data are of high accuracy so that the benefit from the self-labeled training data exceeds the negative effect of the noise contained in the self-labeled training data. This accords with that shown in figure 1 in section 3. The performance of CBCSVM degrades slightly with the increase of *maxIter*. Because soft-constrained k-means cannot cope with cluster bias well, it introduces more noise into the self-labeled training data which further put negative effect on the SVM training. Such noise cumulates in the following iterations, which make the final SVM perform worse than that in ACTC. SemiCCAc outperforms SVM, which accords with the former conclusion that unsupervised learning gives better performance than supervised learning when the size of training data set is extremely small.

[3] http://svmlight.joachims.org/

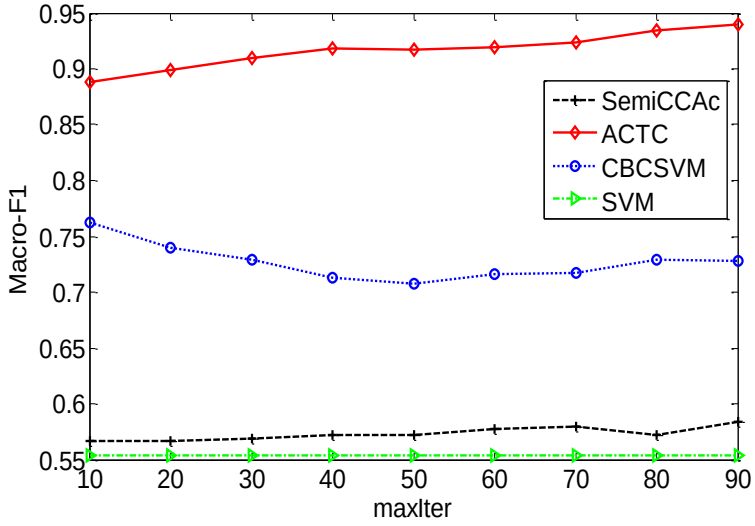

Fig.2. Performance with maxIter on same2

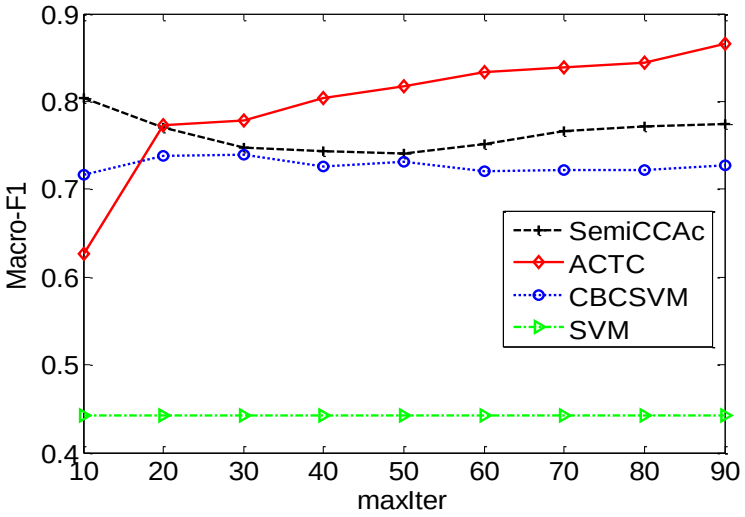

Fig.3. Performance with *maxIter* on Reuters

From figure 3, we can see that ACTC outperforms the other algorithms when *maxIter*>20. This may lies in the fact that the self-labeled training data are unbalanced for each class in SemiCCAc, and that SemiCCAc may filter out useful examples when it copes with the cluster bias, which have relatively larger effect on the final SVM performance when the size of training data set is small. This is expected to be improved by exploring sampling technique on training data set, e.g. over-sampling. On Reuters data set, SemiCCAc

outperforms CBCSVM slightly and significantly outperforms SVM. This indicates that clustering gives better performance than that of SVM when the initial training data set is small.

To evaluate the performance of ACTC with a large range of labeled data, we run the algorithm together with CBCSVM, SVM and SemiCCAc on different percentage of the labeled data on the above two data sets. Figures 4 and 5 give the results. We set *p*=0.5 and *maxIter*=60. We conduct the experiments 30 runs and the average results are given. Training data are randomly sampled in each run.

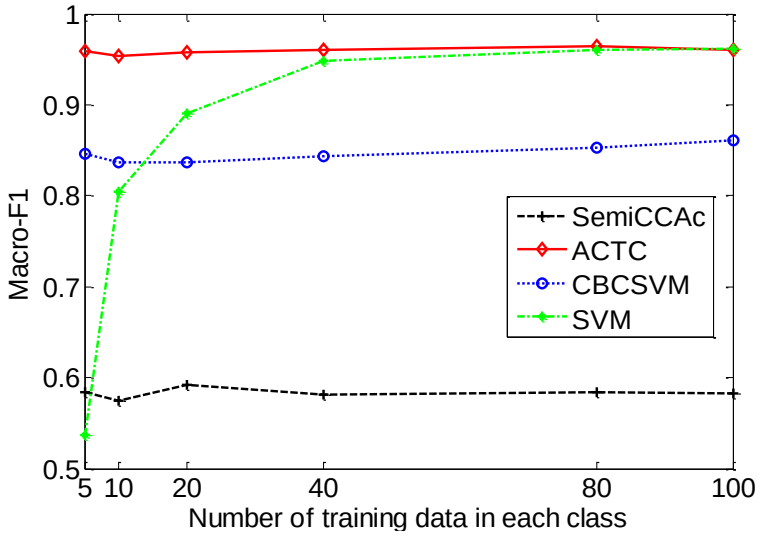

Fig.4. Performance with number of training data on same2

ACTC performs best on the two data sets with all size of training data set. SVM performs worst when the size of training data is 5 for each class. Then its performance improves fast with the increase of training data. SVM outperforms CBCSVM and SemiCCAc when the size of training data set is larger than 20 on same2 and larger than 10 on Reuters. With the increase of training data, the performance of ACTC, CBCSVM, and SemiCCAc grows very slowly. For ACTC and CBCSVM, the reason may be due to the effect of noise contained in the self-labeled training data. Therefore data editing or noise filtering techniques may be helpful to improve the performance. After noise filtering, feature selection and sampling may also be helpful to improve the overall performance. ACTC always significantly outperforms CBCSVM and SemiCCAc, which indicates that our two-stage classification framework is superior to that of CBC, and that the combination of generative model with discriminative model can overcome the shortcomings of both models.

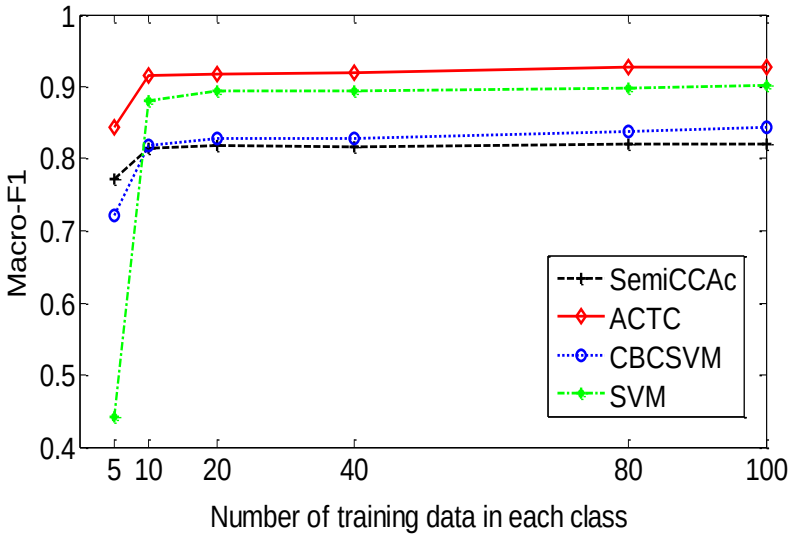

Fig.5. Performance with size of training data set on Reuters

In ACTC and CBCSVM, the parameter *p* determines the number of self-labeled examples in each selection process. Larger value of *p* indicates more examples are self-labeled in each selection, so fewer iterations are needed when the number of self-labeling examples are fixed. But more noise may be introduced into the training data set (please refer to figure 1).

6. Conclusion

This paper presents an active semi-supervised clustering based two-stage classification framework for sparsely labeled text classification. In order to address the cluster bias problem, an active semi-supervised clustering method is proposed. We use a self-training style clustering method to augment the training data set, so that we can convert the challenging problem of sparsely labeled text classification into a supervised one. Therefore supervised classification models can be used, e.g. SVM, and useful techniques for supervised learning can be employed to further improve the performance. The experiments show the superior performance of our method over SVM and CBC (SVM as base learner).

In the future, we plan to evaluate other clustering methods to address the cluster bias problem, e.g. affinity propagation clustering and density based clustering. In terms of noise control, data editing or noise filtering techniques will also be explored. Other directions include investigating the problems of example selection, confidence assessment, and resampling techniques.

Acknowledgment. The authors would like to thank the anonymous reviewers for their useful advice. This work is partially supported by the National Natural Science Foundation of China (No.61127005, No.61170054, No.50708085, and No.50978127), the special scientific research funding of Research Institute of Highway, Ministry of Transport (No.1206030211003), and the Project of Education Department of Jiangxi Province (No.GJJ08415).

References

1. Joachims, T.: Text categorization with support vector machines: Learning with Many Relevant Features. In Proceedings of the European Conference on Machine Learning. Chemnitz, Germany, April 21 – 24, 137--142 (1998).
2. Lewis, D. D.: Naïve Bayes at forty: The independence assumption in information retrieval. In Proceedings of the European Conference on Machine Learning. Chemnitz, Germany, April 21 – 24 (1998).
3. Masand, B., Linoff, G., Waltz, D.: Classifying news stories using memory based reasoning. In Proceedings of the 15th International ACM/SIGIR Conference on Research & Development in Information Retrieval. Copenhagen, Denmark, June 21 - 24, 59-64 (1992).
4. Ng, T. H., Goh, W. B., Low, K. L.: Feature selection, perception learning and a usability case study for text categorization. In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Philadelphia, PA, USA, July 27-31, 1997.
5. Yang, Y. & Liu, X.: An re-examination of text categorization. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, CA, USA, August 15-19, 1999.
6. Joachims, T.: Transductive inference for text classification using support vector machines. In Proceedings of the 16th international conference on machine learning (ICML1999). Bled, Slovenia, June 27-30, 200-209 (1999).
7. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with Co-Training. In Proceedings of the 11th Annual Conference on Computational Learning Theory. Madison, Wisconsin, July 24-26, 92-100 (1998).
8. Nigam, K., McCallurn, A. K., Thrun, S. , Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning, 39(2/3):103-134, 2000.
9. Seeger, M.: Learning with labeled and unlabeled data. Technical report, Edinburgh University, 2001.
10. Slonim, N., Tishby, N.: Document Clustering using Word Clusters via the Information Bottleneck Method. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Athens, Greece, July 24-28, 208--215 (2000).
11. Zeng, H. J., Wang, X. H., Chen, Z., Ma, W. Y.: CBC: Clustering based text classification requiring minimal labeled data. In Proceedings of the 3rd IEEE International Conference on Data Mining. Melbourne, Florida, USA, November 19 – 22, 2003.
12. Yu, H., Yang, J., Han, J.: Classifying large data sets using SVMs with hierarchical clusters. in Proceedings of the 9th ACM SIGKDD 2003, Washington, DC, USA, 2003.
13. Evans, R., Pfahringer, B., Holmes, G.: Clustering and Classification. 7th International conference on information technology in Asia (CITA 11). Sarawak, Malaysia, July 12-13, 1-8 (2011).

14. Kyriakopoulou, A.: (2008). Text classification aided by clustering: a literature review. Tools in Artificial Inteligence, 233-252, 2008.
15. Fung, G., Mangasarian, O.L.: (2001). Semi-supervised support vector machines for unlabeled data classification. Optim. Methods Software, 2001, v15 i1. 29-44.
16. A. Kyriakopoulou, T. Kalamboukis. (2008). Combining clustering with classification for spam detection in social bookmarking systems. in Proceedings of ECML/PKDD Discovery Challenge 2008 (RSDC 2008), Antwerp, Belgium, 2008, pp. 47–54.
17. Kyriakopoulou, A.: Using Clustering and Co-Training to Boost Classification Performance. In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence. Patras, Greece, October 29-31, 325-330 (2007) .
18. Raskutti, B., Ferrá, H., Kowalczyk, A.: (2002). Combining clustering and co-training to enhance text classification using unlabelled data. In Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining. Edmonton, Alberta, Canada, July 23 - 26, 2002.
19. Cai, W., Chen, S., Zhang, D.: A multiobjective simultaneous learning framework for clustering and classification. IEEE Transactions on Neural Networks, 21(2): 185–200, 2010.
20. Qian, Q., Chen, S., Cai, W.: Simultaneous clustering and classification over cluster structure representation. Pattern Recognition, 2011, October 27.
21. Chapelle, O., Weston, J., Scholkopf, B.: Cluster kernels for semi-supervised learning. Advances in Neural Information Processing Systems In NIPS 2002, Vol. 15 (2003), 585-592.
22. Zhou, D., Bousquet, O., Lal, T. N., Weston, J., Scholkopf, B.: Learning with local and global consistency. Advances in Neural Information Processing Systems 16, 321-328, 2004.
23. Keswani, G., Hall, L.O.: Text classification with enhanced semi-supervised fuzzy clustering. Handbook of Fuzzy Computation, 1994, 511-515.
24. Ng, A. Y., Jordan, M. I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. Advances in Neural Information Processing Systems 14, 2002.

Xue Zhang, received the BS degree in electronic engineering from XiDian University, Xian, China, in 1999. She received the MS degree in control theory and control engineering from Southwest University of Science and Technology, Mianyang, China, in 2003, and received the PhD degree in computer science from Southeast University, Nanjing, China, in 2007. From 2008 to the present, she is a postdoctoral fellow in Peking University. Her research interests include data mining and machine learning, with emphasis on the applications to text mining and bioinformatics.

Wangxin Xiao, received the PhD degree in traffic information and control engineering from Southeast University, Nanjing, China, in 2004. From 2005 to 2007, he engaged in postdoctoral research in Wuhan University of Technology. Since 2008 he has been an associate professor in Research Institute of Highway Ministry of Transport. From 2009 to 2011, he was also a postdoctoral fellow in Changsha University of Science and Technology. His research interests include pattern recognition, Intelligent Transport Systems (ITS) and data mining with applications to traffic data.

Received: January 30, 2012; Accepted:December 05, 2012.