DOI: 10.2298/CSIS120202045Z

# Active Semi-supervised Framework with Data Editing

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

[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 Computer Science, Jinggangshan University, Ji'an 343009, China
jane_zhang@pku.edu.cn
[3]School of Traffic and Transportation Engineering, Changsha University of Science and Technology, Changsha 410114, China
wx.xiao@rioh.cn

**Abstract.** In order to address the insufficient training data problem, many active semi-supervised algorithms have been proposed. The self-labeled training data in semi-supervised learning may contain much noise due to the insufficient training data. Such noise may snowball themselves in the following learning process and thus hurt the generalization ability of the final hypothesis. Extremely few labeled training data in sparsely labeled text classification aggravate such situation. If such noise could be identified and removed by some strategy, the performance of the active semi-supervised algorithms should be improved. However, such useful techniques of identifying and removing noise have been seldom explored in existing active semi-supervised algorithms. In this paper, we propose an active semi-supervised framework with data editing (we call it ASSDE) to improve sparsely labeled text classification. A data editing technique is used to identify and remove noise introduced by semi-supervised labeling. We carry out the data editing technique by fully utilizing the advantage of active learning, which is novel according to our knowledge. The fusion of active learning with data editing makes ASSDE more robust to the sparsity and the distribution bias of the training data. It further simplifies the design of semi-supervised learning which makes ASSDE more efficient. Extensive experimental study on several real-world text data sets shows the encouraging results of the proposed framework for sparsely labeled text classification, compared with several state-of-the-art methods.

**Keywords:** sparsely labeled text classification; active learning; semi-supervised learning; data editing

## 1. Introduction

Automatic text classification is of great importance due to the large volume of text documents in many real-world applications. The goal of automatic text classification is to automatically assign documents to a number of predefined categories. A supervised classification model often needs a very large number of training data to enable the classifier's reliable performance. 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 a reliable text classifier.

In order to reduce the effort involved in acquiring labeled examples, there are two major strategies, active learning and semi-supervised learning. The aim of active learning is to select most informative unlabeled examples for manually labeling so that a good classifier can be learned with significantly fewer labeled examples. Active learning has been extensively studied in machine learning for many years and has already been employed for text classification in the past [1-3]. Semi-supervised learning tries to learn a classification model from the mixture of labeled and unlabeled instances, which also has been employed for text classification [4-5]. The fusion of active learning with semi-supervised learning can further bring advantage, thus several combination algorithms have been proposed for text classification [6-7].

Sparsely labeled classification is a special form of classification in which only very few labeled instances are available. It exists in many real-world applications such as content-based image retrieval, online web-page recommendation, object identification and text classification, where the abundant unlabeled instances are available but the labeled ones are fairly expensive to obtain. The sparsity of training data often leads to severe distribution bias between the training data and the unlabeled data (we call it training data bias). It is very difficult to learn a weak useful hypothesis with the extremely few labeled instances. Existing semi-supervised learning and active learning algorithms, which often need quite a number of labeled instances to learn an initial weak useful predictor for further learning, cannot perform well for sparsely labeled text classification [8].

Due to the poor performance of the initially learned hypothesis based on the very few training data, it is unavoidable to contain much noise in the self-labeled instances. Extremely few labeled training data in sparsely labeled text classification aggravate such situation. If such noise could not be identified and removed from the new training data set, they may snowball themselves in the following learning process and thus hurt the generalization ability of the final hypothesis. On the other hand, if such noise could be identified and removed by some strategy, the performance of the active semi-supervised algorithms should be improved. However, such useful techniques of identifying and removing noise have been seldom explored in existing active semi-supervised algorithms, especially using the advantage of active learning to do this. This is one motivation of this work.

In this paper, we propose an active semi-supervised framework with data editing (we call it ASSDE) to improve sparsely labeled text classification. ASSDE conducts in a self-training style process. In order to efficiently integrate active learning, we extend the standard self-training by substituting ensemble classifiers for its single classifier. Furthermore, we introduce a data editing technique into ASSDE by fully utilizing the advantage of active learning. Data editing technique is used to identify and remove the noise contained in self-labeled instances. ASSDE iterates the steps of self-labeling, active labeling and data editing until satisfying some stopping criteria.

The main contributions of this paper may be summarized as follows:

— We propose an active semi-supervised framework with data editing which is more effective and efficient for sparsely labeled text classification compared with state-of-the-art algorithms.
— We carry out a data editing technique to identify and remove the noise contained in self-labeled instances by fully utilizing the advantage of active learning, which is novel and incurs very little computation complexity while improving the classification accuracy.
— We propose a novel parameter ensemble strategy to extend the standard self-training algorithm in order to efficiently integrate active learning while incurring less computation complexity.
— We empirically demonstrate that data editing can simplify the design of semi-supervised learning for efficiency reason, while not degrading the performance.
— We conduct extensive experiments on three benchmark real-world text data sets to evaluate its performance with different parameters.

The rest of this paper is organized as follows. We discuss some of the related work in Section 2. Section 3 describes the algorithm in detail. Section 4 presents the results of the experiments. A short conclusion and future work are presented in Section 5.

## 2. Related Work

A variety of algorithms for text classification have been proposed [1-14], including supervised methods, semi-supervised methods, active methods and the combinations. In the following, we only focus on the study and techniques related to sparsely labeled text classification.

There are several techniques which are beneficial to sparsely labeled text classification. These techniques includes: 1) semi-supervised learning and active learning, 2) transfer learning [12-13], 3) feature extension with semantic concepts [14], 4) clustering aided methods [15-17], 5) data editing [18]. Semi-supervised learning and active learning are two common used techniques to address the problem of insufficient training data. However, they often need quite a number of training data to train a weak useful predictor for further learning. In sparsely labeled classification, it is very challenging, if not impossible, to generate such weak useful predictor, which makes existing

semi-supervised and active learning algorithms cannot be applied. The second technique beneficial to sparsely labeled text classification is transfer learning. It refers to the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task. Transfer learning techniques can be used to improve sparsely labeled classification by transferring the useful knowledge for the problem. The third technique beneficial to sparsely labeled text classification is to utilize the world knowledge (e.g. Wiki, WordNet). By using the world knowledge, the feature representation of an instance is enhanced by semantic concepts which can weaken the feature sparsity in some degree. Clustering aided methods beneficial to sparsely labeled classification including both expanding the training data from unlabeled data [15] and augmenting the data set with new features [17]. Another technique beneficial to sparsely labeled text classification is data editing. It can be explored to indentify and remove the noise contained in self-labeled instances. In this paper, we address the problem of sparsely labeled text classification by active semi-supervised learning with data editing.

In sparsely labeled classification, the generalization ability of the hypothesis learned on the initial training data is often very poor. Thus there may contain much noise in the self-labeled instances. In self-training style algorithms, the early introduced noise by semi-supervised learning may snowball themselves, which often makes the final hypothesis of very poor performance. If such noise could be identified and removed by exploring some useful techniques, the classification accuracy should be improved. Data editing technique could be used for this end. In conventional studies, data editing aims to remove noisy instances from the original training data set with the goal to improve classification accuracy by producing smooth decision boundaries. A new self-training style algorithm, SETRED, is proposed in [19] by introducing a data editing technique to the self-training process to filter out the noise in the self-labeled instances. SETRED outperforms the standard self-training, which indicates that the performance of semi-supervised learning can be further improved by introducing proper data editing technique. This paper shows that it is encouraging to fuse active learning with data editing to identify and remove the noise in the self-labeled instances. It incurs very little computation complexity while improving the classification accuracy.

Although sparsely labeled text classification is a very significant problem in many real-world applications, there have been very limited researches on it. A clustering based classification method, CBC [15], is one such work. It combines transductive support vector machines (TSVM) with k-means and iterates these two steps alternatively. Although clustering can help to overcome the sparsity and the training data bias, CBC has a very high computation complexity because both TSVM and k-means are very time-consuming for sparsely labeled high dimensional text classification. Based on kernel canonical component analysis, OLTV (learning with One Labeled example and Two Views) [20] and ALESLE (Active Learning with Extremely Sparse Labeled Examples) [21] algorithms have been proposed for sparsely labeled classification. While OLTV works in semi-supervised setting, ALESLE works in

active setting. OLTV and ALESLE require two sufficient views, which is not practical for many real-world applications.

Phan et al. 2008 [22] propose a general framework for building classifiers that deal with short and sparse text & Web segments by making the most of hidden topics discovered from large-scale data collections. This framework can be viewed as a semi-supervised learning technique, but it is flexible in that the universal data are not necessary to have the same format as the labeled training or future unseen data. Cai et al. 2003 [23] propose a framework for text categorization which attempts to analyze topics from both training and test data using probabilistic latent semantic analysis (PLSA) and uses both the original data and resulting topics to train two different weak classifiers for boosting. Xu et al 2008 [24] propose a web-assisted text categorization framework which automatically identifies important keywords from the available labeled documents to form the queries and then uses search engines to retrieve from the Web a multitude of relevant documents. These retrieved documents are then exploited by a semi- supervised framework.

## 3. ASSDE Framework

### 3.1. Problem Description and Notation

Let $D=L \cup U$ denote the set of instances in a $p$-dimensional Euclidean space $R^p$, where $L=\{<x_i,y_i>\}_{1\le i\le l}$ is the set of labeled instances and $U=\{x_i\}_{l+1\le i\le l+u}$ the set of unlabeled ones. Here $y_i$ is the class label of instance $x_i$ and $l<<u$. This paper only considers single-label classification that exact one label should be assigned to each instance in $D$. The set of classes is denoted by $C=\{c_r\}_{1\le r\le |C|}$ and $|C|$ is the cardinality of $C$ and $r$ is an integer. Each instance in $L$ has been labeled or assigned to one class in $C$, while the class label of each instance in $U$ is unknown and needs to be determined.

### 3.2. ASSDE Framework

The aim of this work is to improve sparsely labeled text classification by introducing data editing technique into active semi-supervised framework. As we know, one of the problems of semi-supervised learning is its learning efficiency. To make the framework more efficient, we also expect to simplify the design of semi-supervised learning by fully utilizing the advantage brought by the fusion of active learning with data editing. We show empirically that data editing can make up for the deficiency of simplifying the design of semi-supervised learning. Table 1 gives the Pseudo-code description of ASSDE.

**Table 1.** Pseudo-code describing ASSDE algorithm

```
Algorithm: ASSDE
Input: the labeled set L, the unlabeled set U, cpNum, MaxcpIter, n
Output: the full labeled set D=L ∪U
Progress:
  Offline:
    Computing distance matrix(L ∪ U)
  Online:
    h_i ← Learn(L),i=1,2,...
    Iter=0
    L' ← ∅, SL' ← ∅, SL ← ∅
    Repeat until no data in U can be put into SP or U = ∅
      (SP,CP)=partition(U,h_1,h_2,…)
      SL' = choose(SP, n)
      SL = SL ∪ SL'
      U = U − SL'
      If Iter<maxcpIter
        L' = BMAL(CP, cpNum)
        L = L ∪ L'
        U = U − L'
        h_i ← Learn(L),i=1,2,...
        SL=recheck(SL,h_1,h_2,…)
        Iter= Iter+1
      End
      h_i ← Learn(L ∪ SL),i=1,2,...
    End
    If U ≠ ∅ then for each instance in U,
      c=majority voting(h_1,h_2,…)
    End
```

ASSDE works as follows. Firstly ensemble classifiers are trained on the initial training data set *L*. The trained ensemble classifiers are used to predict the label of each instance in *U*. According to the labels predicted by the ensemble classifiers, instances in *U* are partitioned into two sets, that is, contention points set *CP* and consistent points set *SP*. *CP* contains the instances whose labels predicted by the ensemble classifiers are inconsistent, while *SP* consists of the instances which have consistent predicted labels. Secondly, *n* most confident examples, say $SL'$, are selected from *SP* and labeled with their predicted labels. Then *cpNum* instances, say $L'$, from *CP* are selected by a batch mode active learning algorithm for manually labeling. Due to the small size of *L*, the generalization ability of the hypothesis learned by ensemble learning may be poor. Consequently, $SL'$ may contain much noise which will hurt the generalization ability of the final hypothesis with the accumulation of such noise in the following self-training processes. Therefore, we employ a data editing technique to identify and remove the noise. We use

the ensemble classifiers retrained on $L \cup L'$ to predict the labels of self-labeled instances in $SL'$. For each instance in $SL'$, if the newly predicted label is inconsistent with its current label, then it will be removed from $SL'$ and thrown into unlabeled data set *U* again. Now the training data set consists of instances with ground-truth labels and self-labeled instances (denoted by *SL*).

ASSDE conducts in a self-training style process in which the steps of self-labeling, active labeling and data editing are iterated alternatively. After the completion of active learning, all the self-labeled instances labeled in former iterations will be rechecked again. ASSDE iterates the self-training style process until almost all the unlabeled instances are labeled with high confidence. If there is any instance in *U*, we use the majority voting strategy to label it.

### 3.3. Ensemble Strategy

Several useful ensemble techniques have been proposed, such as the well-known training data resampling [25] and input feature resampling [26]. However, we only use a simple ensemble strategy (we construct the ensemble classifiers using k-nearest neighbor (kNN) with different *k* parameter) in ASSDE for two main reasons. One reason is that the well-known ensemble techniques are either not suitable for sparsely labeled text classification or incurring large computation/storage complexity. Intuition and empirical experiments indicate that training data resampling technique is not suitable for sparsely labeled case, since it may further aggravate the sparsity of training data for each component classifier. Input feature resampling technique may be helpful, but it can increase the computation and storage complexity because we have to compute the nearest neighbors of an instance for each component classifier and store the corresponding distance matrix. Furthermore, our empirical experiments with FASBIR (Filtered Attribute Subspace based Bagging with Injected Randomness) [27] also indicate that ensemble learning itself can hardly address the sparsely labeled classification problem well. Based on this fact, the aim of ensemble strategy in ASSDE is to make efficient integration of active learning and the overall efficiency. This is the next reason that we use a simple ensemble strategy in ASSDE.

From table 1 we can see that the efficiency of ASSDE is mainly determined by the ensemble part. In order to improve the efficiency of ASSDE, we can simplify the design of ensemble part while not greatly degrading the overall accuracy. We use kNN as the base learner and construct the ensemble classifiers using kNN with different *k* parameter. The component classifiers are trained in parallel training style. Therefore it can generate the ensemble predictions by only computing the maximal *k* nearest neighbors for an instance and it only needs to store one distance/similarity matrix for all ensemble classifiers. Our ensemble strategy only has the similar computation and storage complexity with that of one kNN with the maximal *k* parameter in the component classifiers. This is the main reason that we select kNN as the base classifier and use the simple ensemble strategy. We can also compute the

distance/similarity matrix beforehand to further make the online learning and classifying procedures efficient, which make kNN an efficient base classifier for algorithms conducted in iterated mode.

### 3.4. Batch Mode Active Learning

The key of batch mode active learning (BMAL) is to ensure the selected instances of both informativeness and diversity. BMAL method [3] based on farthest-first traversal (we call it BMAL_FFT) is based on the intuition that for two examples, the larger the distance between them, the smaller redundancy the information they provide.

BMAL_FFT works as follows. First, it selects an instance x from *CP* randomly or according to its uncertainty for the learning model, and adds x to query set *Q*. Then it selects the next instance $x_i$ according to equation (2) and adds $x_i$ to *Q*. BMAL_FFT repeats the above selection procedure until the needed number of instances has been selected.

$$\mathrm{d}(\mathrm{x}_i, Q) = min_{\mathrm{y}\in Q}\|\mathrm{x}_i - \mathrm{y}\| \quad (1)$$

$$\mathrm{x}_i = \mathrm{argmax}_{\mathrm{x}_i\in CP}\mathrm{d}(\mathrm{x}_i, Q) \quad (2)$$

BMAL_FFT is a global search method which may be not efficient for very large-scale text classification problem. In this paper, BMAL_FFT selects instances from *CP* set, which has a much smaller search space and whose instances are more informative than the whole unlabeled data set *U*.

### 3.5. Data Editing Strategy

The sparsity of training data in sparsely labeled classification often makes the generalization ability of the initial hypothesis very poor. There may contain much noise in the self-labeled data set *SL* because the classifiers may incorrectly assign labels to some unlabeled instances. Such noise may accumulate in the following iterations which will hurt the generalization ability of the final hypothesis. It is obvious that if the mislabeled instances in *SL* could be identified and removed, especially in the early iterations, the learned hypothesis is expected to be better. This is the basis that we introduce data editing technique into the proposed framework to identify and remove the mislabeled instances.

After each active learning process, the training data with ground-truth labels increase. In general terms, the classifiers trained on the enlarged training data set will generate more accurate hypothesis. It may be helpful using this hypothesis to identify the noise contained in the self-labeled data set labeled by former less accurate hypotheses. Our data editing strategy is based on this intuition. It works as follows. After each active learning process, the ensemble classifiers are retrained on the enlarged training data set $L \cup L'$. Then they are used to predict the label of each instance in *SL*. If any inconsistency exists

between the newly predicted label and its current label for an instance, the instance will be removed from *SL* and added to *U* again.

In ASSDE, the introduction of data editing technique provides many chances for self-labeled instances to mend their ways, which is different from the traditional semi-supervised learning. Furthermore, we carry out data editing technique by fully utilizing the advantage of active learning, which incurs very little computation complexity while improving the classification accuracy. This is novel in active semi-supervised learning community, according to our knowledge.

We think that data editing technique should be very useful especially in sparsely labeled text classification since the extremely few labeled training data make the mislabeling unavoidable. Data editing technique makes the mistakes made in earlier stages not as severe as that in traditional semi-supervised learning algorithms. Based on this fact, we can simplify the design of the module that determines the algorithm's overall efficiency, and use data editing technique to make up for the deficiency. In ASSDE, we design the ensemble strategy just based on this mind.

## 4. Experiments

### 4.1. Data Sets

For a consistent evaluation, we conduct our empirical experiments on three benchmark data sets, 20NewsGroups, Reuters-21578 and email spam filtering data set. The details of data sets are given in table 2.

20 Newsgroups is one famous Web-related data collection. From the original 20 Newsgroups data set, same-2, consisting of 2 very similar newsgroups (comp.windows.x, comp.os.ms -windows), and diff-2, consisting of 2 very different newsgroups(alt.atheism and comp.windows.x), are used to evaluate the performance of the algorithms on data sets with different separability. Same-2 and diff-2 both contain 2000 instances, 1000 for each class. We use Rainbow software[1] to preprocess the data (removing stop words and words whose document frequency less than 3, stemming) and we get 7765 and 8599 unique terms for same-2 and diff-2, respectively. Then terms are weighted with their TFIDF (Term Frequency-Inverse Document Frequency) values.

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

[1] http://www.cs.cmu.edu/~mccallum/bow/
[2] http://ews.uiuc.edu/~jinggao3/kdd08transfer

The email spam data set, released by ECML/PKDD 2006 discovery challenge, contains a training set of publicly available messages and three set of email messages from individual users as test sets. The algorithms have similar performance on different email spam data sets in LWE[2], therefore we only show the experimental results of test1.svm for the limited space. test1.svm contains 2500 messages and 83636 unique terms.

**Table 2**.The details of data sets

| Data sets | #classes | #total instances | #features | #training instances for each class |
|---|---|---|---|---|
| same-2 | 2 | 2000 | 7765 | 5 |
| diff-2 | 2 | 2000 | 8599 | 5 |
| Reuters | 2 | 1239 | 6889 | 5 |
| Spam | 2 | 2500 | 83636 | 5 |

### 4.2. Performance Evaluation

Macro_*F*1 is used as the performance measurement. *F*1 metric is defined as $F1 = 2 \cdot P \cdot R/(P + R)$, where *P* and *R* are precision and recall for a particular class. *F*1 metric takes into account both the precision and the recall, thus is a more comprehensive metric than either precision or recall when separately considered.

Macro_*F*1 is a measurement which evaluates the overall performance of the classification model. Macro_*F*1 is defined in equ. (3), where $F_1^i$ is the *F*1 value of class *i*.

$$\text{Macro\_}F1 = \frac{1}{|C|}\sum_{i=1}^{|C|} F_1^i \tag{3}$$

### 4.3. Experimental Results and Analysis

In ASSDE, one important parameter is *n*, which affects both the effectiveness and efficiency of the proposed method. *cpNum* and *MaxcpIter* are pairwise parameters which also affect the effectiveness and efficiency of the proposed method. In the following, we mainly conduct experiments with respect to parameters *n*, *cpNum* and *MaxcpIter*, to study their influence on the performance of the proposed method. At the same time, we conduct experiments with several degenerated variants of ASSDE to see the contribution of each component technique. Furthermore, we compare our method with those of several state-of-the-art algorithms both in effectiveness and in efficiency.

In the following experiments, the ensemble size is 3 and we set *k*=1, 3, 5 for the three component kNN classifiers in ASSDE and its degenerated variants. We use random strategy to select the first instance in the batch mode active learning. We conduct each experiment 40 runs and the average results are

given. In each run, the algorithms perform on the same randomly chosen training data set which has 5 positive labeled instances and 5 negative labeled instances. The runtime given below for all algorithms is the average online time.

**Robustness with different parameter *n*.** Since ASSDE integrates several techniques, its two degenerated variants can be easily derived. First, if only ensemble learning and self-training are employed, then EnST (ensemble style self-training) algorithm is obtained. That is, EnST algorithm substitutes ensemble classifiers for one base classifier in standard self-training. Second, if EnST algorithm is augmented by active learning (BMAL_FFT), then AcEnST algorithm is obtained. Note that, although ASSDE integrates ensemble learning, active learning and data editing together, we can only obtain its two degenerated variants, EnST and AcEnST. This is because of the dependent relationships among the component techniques in ASSDE. From table 1 we can find that active learning is based on the ensemble learning and data editing is based on the active learning.

In this subsection, we conduct experiments for ASSDE and its degenerated variants with different parameter *n* in order to see the contribution of each component technique and the robustness of the algorithms with parameter *n*. The standard self-training algorithm (ST) is taken as the baseline and it also refers to kNN as its base classifier. Here we set *cpNum*=30, *MaxcpIter*=6 for algorithms with batch mode active learning.

Figures 1, 2, 3, and 4 give the Macro_*F*1 performance with different parameter *n* for all algorithms on four data sets. For the standard self-training (ST) algorithm, we only give the performance with k=3 since it has similar performance with k=3 and k=5, but it performs relatively worse with k=1.

Figure 1 presents the Macro_F1 performance of the algorithms with different parameter *n* on same-2. It could be found that ASSDE significantly outperforms the other three algorithms, which verify the usefulness of the fusion of active semi-supervised learning with data editing technique. Furthermore, ASSDE is more robust than the other three algorithms to parameter n. EnST outperforms ST slightly with most parameter *n*, which accords with our former analysis that the simple ensemble strategy in ASSDE is designed to make efficient integration of active learning and to achieve overall efficiency while without degrading the performance. AcEnST outperforms EnST with all parameter *n*, which verify the usefulness of active learning. Note that, ASSDE performs better with larger values of parameter *n*, which makes it very efficient since larger *n* means lower computation complexity (less iterations).

Figure 1 also shows that the algorithms except ASSDE perform better with the increase of parameter *n* first, then worse with the increase of parameter *n*. We think this is due to the paradox that larger parameter *n* makes larger training data set available for retraining the classifiers in next iteration which is very useful for sparsely labeled classification especially in the earlier iterations, but larger parameter *n* also means that more noise may be introduced into training data set which will hurt the learning results in the following iterations. Therefore there should be a balance point on which the classification model achieves its best performance. Different algorithms achieve this balance point

with different parameter *n*. For example, ST achieves its balance point with about *n*=250, EnST and AcEnST with about *n*=150. It seems that data editing technique could delay such balance point (with the increase of *n*) by identifying and removing the noise contained in self-labeled training data.

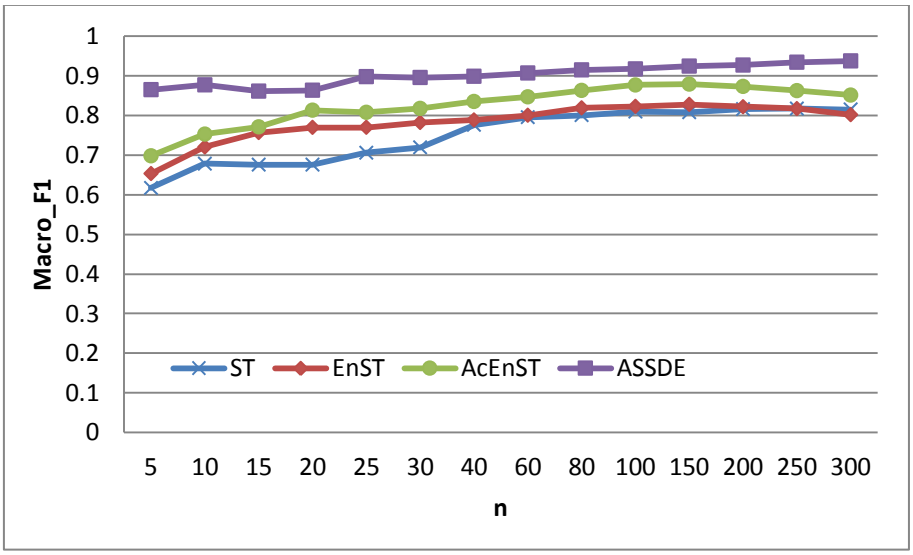


**Fig.1.** Macro_F1 performance with different parameter *n* on same-2

Figure 2 presents the Macro_*F*1 performance of the algorithms with different parameter *n* on diff-2. From figure 2 it could be found that ASSDE outperforms the other three algorithms with all parameter *n* and it is very robust to parameter *n*. EnST and ST perform similarly. AcEnST outperforms EnST and ST, and it achieves the best performance with n=5. For ST and EnST, there also exists the balance point phenomenon. ST achieves its best performance with about n=80, EnST with about n=60. The results in figure 2 also accords with our analysis in former sections.

Figure 3 gives the Macro_*F*1 performance of the algorithms with different parameter *n* on Reuters. ASSDE outperforms the other three algorithms with all parameter *n* and it is more robust to parameter *n*. EnST is outperformed by ST with small *n*, but it outperforms ST when n>80. AcEnST outperforms ST and EnST with all parameter *n* and its performance degrades with the increase of *n* when *n*>30. ASSDE achieves its best performance with about *n*=80, ST with about *n*=15, EnST with about *n*=10, and AcEnST with about *n*=20. From the balance point of the four algorithms, we can see that ASSDE is most efficient for its least iterations, which must benefit from data editing technique.

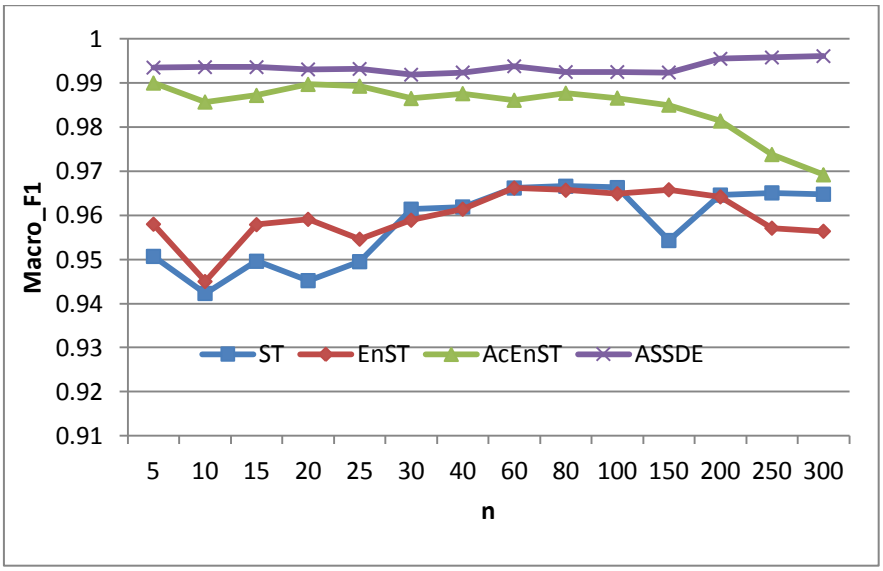


**Fig.2.** Macro_*F*1 performance with different parameter *n* on diff-2

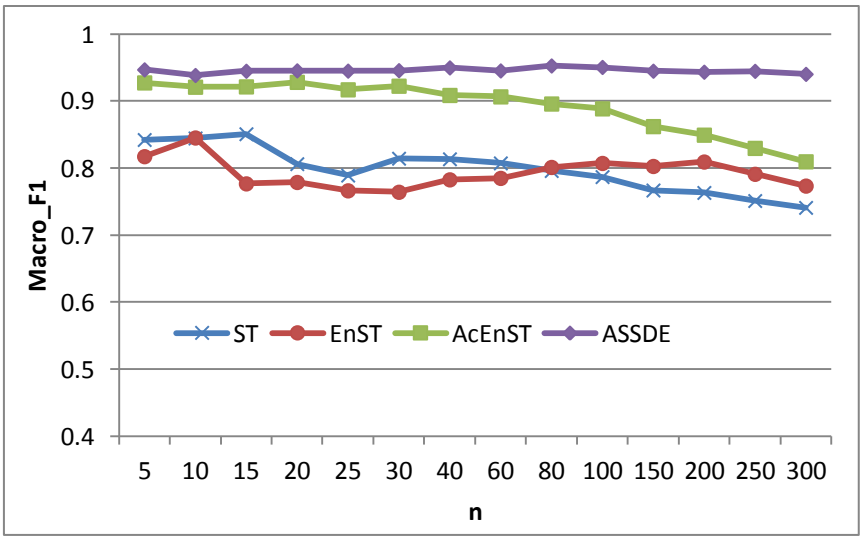


**Fig.3.** Macro_F1 performance with different parameter *n* on Reuters

Figure 4 presents the Macro_*F*1 performance of the algorithms with different parameter *n* on Spam data set. It could be found that ASSDE outperforms ST and EnST with all parameter *n*, and outperforms AcEnST when *n*>60. ASSDE ties with AcEnST when *n*<60, but it is more robust with parameter *n* than other

three algorithms. EnST performs slightly better than ST with almost all parameter *n*. AcEnST significantly outperforms ST and EnST with all parameter *n*. It seems that active learning plays a leading role for performance improvement in ASSDE on Spam data set. ASSDE achieves its best performance with about *n*=200, AcEnST with about *n*=25 or *n*=40, EnST with about *n*=80, and ST with about *n*=100.

In general, from figures 1 to 4, we can conclude that active learning in ASSDE is always beneficial to improve the overall performance, and that data editing technique plays a key role in the robustness and improving the performance for ASSDE. Furthermore, data editing technique makes ASSDE more efficient since it can make ASSDE achieve its best performance with larger values of *n* (correspondingly less iterations). It seems that data editing technique brings larger advantage for data set of lower separability (e.g. same-2). This may be due to the fact that there should be more noise in self-labeled instances for data set of lower separability. Thus data editing technique can greatly improve the overall performance by identify and remove such noise contained in self-labeled training data. This accords with our intuition.

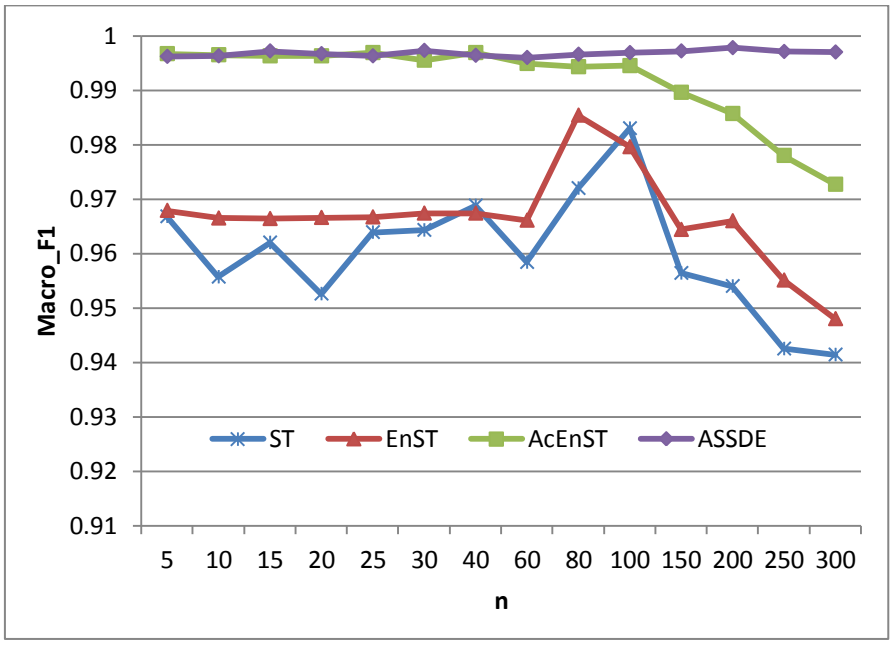


**Fig.4.** Macro_F1 performance with different parameter *n* on Spam

**Robustness with cpNum & MaxcpIter.** Parameters of *cpNum* and *MaxcpIter* determine the amount of manual effort involved. In general, for an active learning algorithm, the more the manual effort involved, the better the performance it achieves. The aim of active learning is to reduce the effort involved without degrading the performance. In ASSDE, the performance will be improved with the increase of the product of *cpNum* and *MaxcpIter*, which is more intuitive. Therefore we only conduct experiments in the case of *cpNum*MaxcpIter*=constant, while *cpNum* and *MaxcpIter* may take different values.

The following experiments are also conducted on the basic training data set which contains 5 training data for each class and is sampled at random in each run. We test the performance of ASSDE with *cpNum*=10, 20, 30, 60 and 90, correspondingly *MaxcpIter*=18, 9, 6, 3, and 2. Since the iterations of ASSDE are determined by parameter *n*, the iterations may be less than *MaxcpIter* with larger value of parameter *n*. That is, in the following experiments, ASSDE can at most actively label 180 instances in each run. We set *n*=300.

Figure 5 shows the performance of ASSDE with *cpNum* on four data sets. In the case of *cpNum* MaxcpIter*=constant, the performance declines slightly with the increase of *cpNum*, and at the same time the runtime also declines (please see figure 6). We think this is because that the redundancy among the actively selected instances increases with the increase of *cpNum*. In general, *cpNum*=30 (correspondingly *MaxcpIter*=6) is best for ASSDE when considering both the effectiveness and the efficiency on the four data sets.

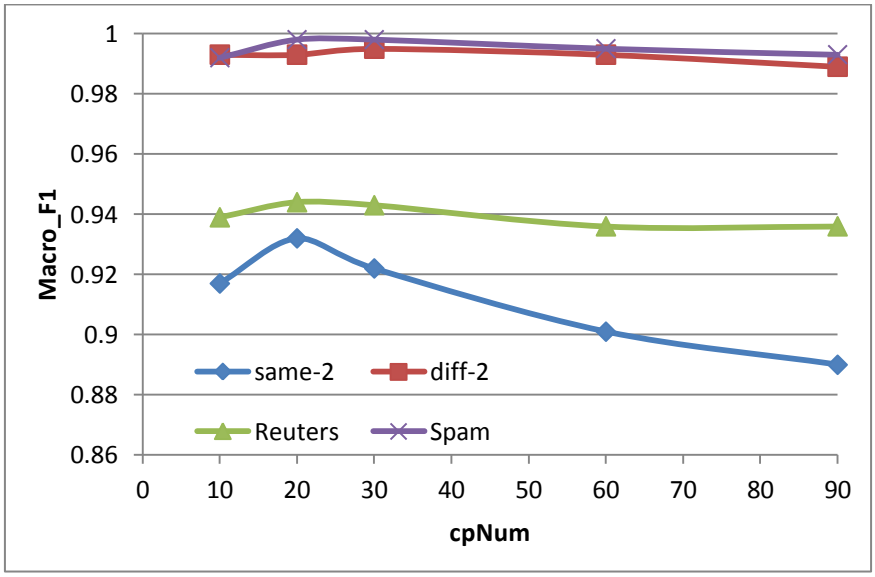


**Fig.5.** Performance with *cpNum* on four data sets

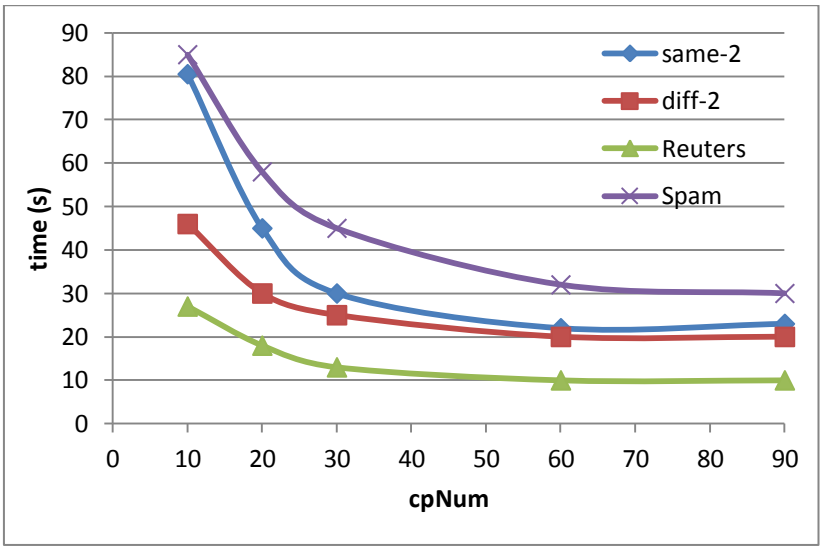


**Fig.6.** Runtime with cpNum on four data sets

**Performance Comparison.** To evaluate the performance of ASSDE with the increase of labeled training data, we conduct the following experiments to compare the performance of ASSDE, Support Vector Machines (SVM, one of the most successful supervised algorithm) and TSVM (one of the most successful semi-supervised algorithm) with different labeling rate. We set *n*=300, *cpNum*=30 and *MaxcpIter*=6 for ASSDE. Since the iterations of ASSDE are determined by parameter n, in the following experiments, ASSDE can at most actively label (selected by active learning and labeled manually by an expert) 180 instances in each run.

For fair comparison between the active algorithm (e.g. ASSDE) and non-active algorithm (e.g. SVM and TSVM), we conduct experiments with two training data sets, that is, the basic training data set (5 training data for each class) for all three algorithms and the extended training data set (basic training data set extended with cpNum*MaxcpIter randomly selected training data) for SVM (denoted as SVM*) and TSVM (denoted as TSVM*). The SVMlight package[3] is used in our experiments for the implementation of SVM and TSVM using default configurations.

Since ASSDE performs very well on diff-2 and Spam (see figures 2 and 4), there is very little room for it to improve performance with the increase of training data size. Therefore, we only conduct the experiments on same-2 and Reuters. For each figure, the x-axis represents the number of training data in each class in basic training data set. Figures 7 and 8 give the performance comparison results.

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

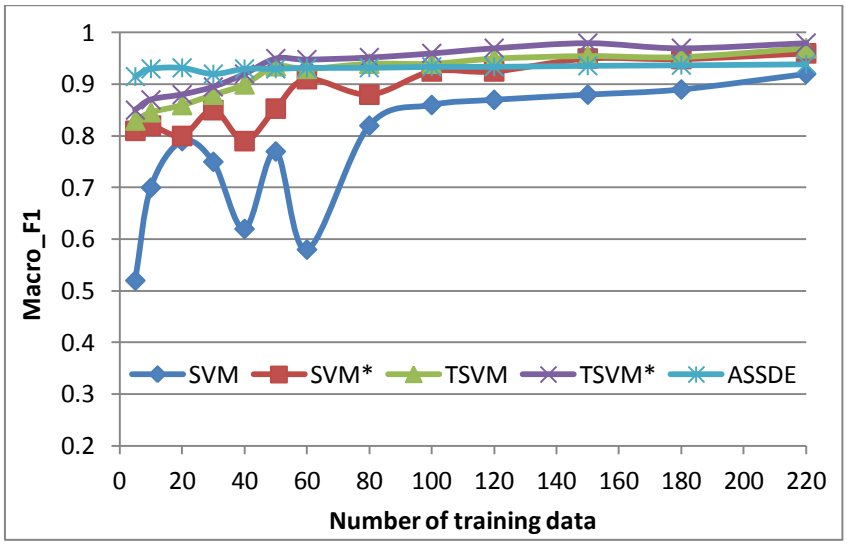


**Fig.7.** Performance comparison on same-2

From figure 7, we can see that ASSDE outperforms other algorithms when training data are less than 40 (4%) on same-2. From figure 8, we can see that ASSDE outperforms other algorithms with all training data size on Reuters. Compared with other algorithms, ASSDE performs better and more robust for sparsely labeled text classification.

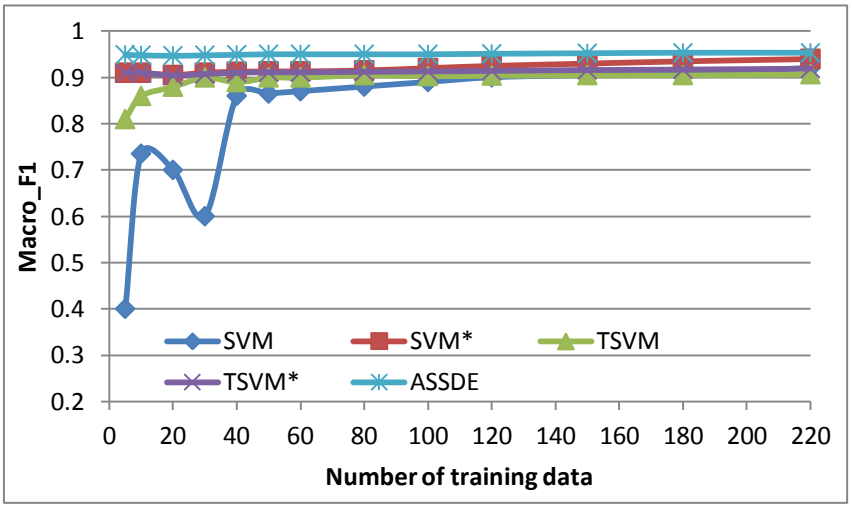


**Fig.8.** Performance comparison on Reuters

## 5. Conclusion

In this paper, an active semi-supervised framework with data editing is proposed to improve the performance of sparsely labeled text classification. The aim of data editing in ASSDE is to identify and remove the noise contained in self-labeled training data and thus to improve the overall performance. Our basic consideration is to implement data editing technique by fully utilizing the advantage of active learning in order to incur less computation complexity while improving the accuracy. At the same time, we expect to simplify the design of key component which determines ASSDE's efficiency and use data editing to make up for the deficiency. Therefore we use a very simple but efficient ensemble strategy in ASSDE. Extensive experiments on four text data sets show that data editing is a very useful technique for improving the performance of sparsely labeled text classification, and it makes the algorithm more efficient. This accords with our expectation.

For future work, we will explore more suitable active learning and data editing techniques which may further improve the performance of sparsely labeled text classification. More efficient and effective ensemble strategy for sparsely labeled text classification will be another research direction. Moreover, we will further explore new techniques to cope with the training data sparsity and training data bias for sparsely labeled text classification, e.g. semantic feature extension and clustering aided techniques.

**Acknowledgment.** The authors would like to thank Dr. Xu Zhu and the anonymous reviewers for their critical advice. This work is partially supported by the National Natural Science Foundation of China (No.61127005, No.61133010, 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. Liere, R., Tadepalli, P.: Active learning with committees for text categorization. In Proceedings 14th Conference of the American Association for Artificial Intelligence (AAAI). MIT Press, Providence, Rhode Island, 591–596 (1997).
2. Steven, C. H. Hoi, Jin, R., Lyu, M. R.: Large-scale text categorization by batch mode active learning. In Proceedings of the 15th international conference on World Wide Web. Edinburgh, Scotland, May 23-26, 633-642 (2006).
3. Zhang, X., Zhao, D. Y., Chen, L.W., Min, W.H.: Batch Mode Active Learning based Multi-View Text Classification. In Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, China, Aug.14-16, 2009.
4. 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).
5. Nigam, K., Mccallum, A. K., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 39(2-3): 103-134, 2000.

6. McCallum, A., Nigam, K.: Employing EM and pool-based Active Learning for text classification. In Proceedings of the 15th International Conference on Machine Learning. Madison, Wisconsin, USA, July 24-27, 350–358 (1998).
7. Gu, P., Zhu, Q. S., Zhang C.: A multi-view approach to semi-supervised document classification with incremental naïve bayes. Computers and Mathematics with Applications, 57(6):1030-1036, 2009.
8. Zhou, Z.H., Zhan, D.C., Yang, Q.: Semi-supervised learning with very few labeled training examples. Association for the advancement of artificial intelligence, 2007.
9. Zhuang, D., Zhang, B.Y., Yang, Q., Yan, J., Chen, Z., Chen, Y.: Efficient text classification by weighted proximal SVM. In Proceedings of the Fifth IEEE International Conference on Data Mining. Houston, Texas, USA, November 27 - 30 538-545 (2005).
10. Liu, T., Chen, Z., Zhang, B.Y., Ma, W.Y., Wu, G.Y.: Improving text classification using local latent semantic indexing. In Proceedings of the Fourth IEEE International Conference on Data Mining. Brighton, UK, November 01–04, 162-169 (2004).
11. Yang, Y., Liu, X.: A re-examination of text categorization methods. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. University of California at Berkeley, USA, August 15-19, 42-49 (1999).
12. Dai, W., Xue, G., Yang, Q., Yu, Y.: Transfering Naïve Bayes Classifiers for Text Classification. Association for the Advancement of Artificial Intelligence, 2007, 540-545.
13. Banerjee, S.: Boosting Inductive Transfer for Text Classification Using Wikipedia. In Proceedings of the sixth international conference on machine learning and applications. Kingsgate Marriott, December 13-15, 148-153 (2007).
14. Wang, P., Domeniconi, C.: Building Semantic Kernels for Text Classification using Wikipedia. In Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. Las Vegas, NV, USA, August 24-27, 713-721 (2008).
15. 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.
16. Raskutti, B., Ferrá, H., Kowalczyk, A.. 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, 620-625 (2002).
17. Raskutti, B., Ferrá, H., Kowalczyk, A.: Using Unlabelled Data for Text Classification through Addition of Cluster Parameters. In Proceedings of the Nineteenth International Conference on Machine Learning. Sydney, Australia, July 8-12, 514 – 521 (2002).
18. Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics, 2:408-420, 1972.
19. Li, M., Zhou, Z.H.: SETRED: Self-training with editing. In Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05), Hanoi, Vietnam, May 18-20, 611-621 (2005).
20. Zhou, Z. H., Zhan, D. C., Yang, Q.: Semi-supervised learning with very few labeled training examples. Association for the advancement of artificial intelligence, 2007.
21. Sun S. L.: Active learning with extremely sparse labeled examples. In Proceedings of Neural Information Processing Systems Workshop on Learning from Multiple Sources. Vancouver, British Columbia, Canada, December 8-11, 2008.

22. Phan, X. H., Nguyen, L. M., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In Proceedings of International World Wide Web Conference. Beijing, China, April 21 – 25, 91-100 (2008).
23. Cai, L., Hofmann, T.: Text categorization by boosting automatically extracted concepts. Proc. ACM SIGIR, Toronto, Canada. July 28 - August 1, 2003.
24. Xu, Z., Jin, R., Huang, K., Lyu, M. R., King, I.: Semi-supervised text categorization by active search. In Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM 2008). Napa Valley, California, October 26-30, 1517-1518 (2008).
25. Breiman L.: Bagging predictors. Machine Learning, 24(2):123–140, 1996.
26. Bryll, R., Ricardo, G. O., Quek, F.: Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition 36(2003):1291-1302.
27. Zhou, Z. H., Yu, Y.: Ensemble local learners through multimodal perturbation. IEEE transactions on systems, man, and cybernetics-Part B: Cybernetics, 2005, 35(4): 725-735.

**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: February 02, 2012; Accepted: November 29, 2012.*