TCSST: Transfer Classification of Short & Sparse Text Using External Data

Guodong Long, Ling Chen, Xingquan Zhu, Chengqi Zhang
QCIS[*], University of Technology Sydney, Australia
Guodong.Long@student.uts.edu.au,
{Ling.Chen, Xingquan.Zhu, Chengqi.Zhang}@uts.edu.au

ABSTRACT

Short & sparse text is becoming more prevalent on the web, such as search snippets, micro-blogs and product reviews. Accurately classifying short & sparse text has emerged as an important while challenging task. Existing work has considered utilizing external data (e.g. Wikipedia) to alleviate data sparseness, by appending topics detected from external data as new features. However, training a classifier on features concatenated from different spaces is not easy considering the features have different physical meanings and different significance to the classification task. Moreover, it exacerbates the "curse of dimensionality" problem. In this study, we propose a transfer classification method, TCSST, to exploit the external data to tackle the data sparsity issue. The transfer classifier will be learned in the original feature space. Considering that the labels of the external data may not be readily available or sufficiently enough, TCSST further exploits the unlabeled external data to aid the transfer classification. We develop novel strategies to allow TCSST to iteratively select high quality unlabeled external data to help with the classification. We evaluate the performance of TCSST on both benchmark as well as real-world data sets. Our experimental results demonstrate that the proposed method is effective in classifying very short & sparse text, consistently outperforming existing and baseline methods.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*text analysis*; I.2.6 [**Artificial Intelligence**]: Learning

[*]QCIS: centre for Quantum Computation & Intelligent Systems

General Terms

ALGORITHMS, PERFORMANCE, EXPERIMENTATION

Keywords

Short & Sparse Text Mining, Classification, Transfer Learning, Wikipedia, External Data

1. INTRODUCTION

Text categorization, one of the most important tasks of text mining, has been studied intensively in the past decade. Various algorithms (e.g. Naive Bayes, Maximum Entropy) have been applied to text classification and proved to be effective on different benchmark datasets (e.g. 20Newsgroups, Reuters-21578). Due to the rapid advancement of information technologies and web applications, there has been an increasingly large amount of *short & sparse text* data available online, such as micro-blogs, web search snippets, forum messages and product reviews etc. Consequently, mining short & sparse text to derive high quality knowledge will benefit many information retrieval and web applications.

Briefly, short & sparse text refers to a data set where each text instance is short, consisting of from a dozen words to a few sentences. Moreover, two instances usually do not share enough word co-occurrence. These new features render traditional text mining tools ineffective, motivating the development of novel methodologies. For example, for the research tasks of measuring similarity between short texts and clustering short texts, methods have been proposed to overcome the data sparsity by expanding original data with information or context extracted from external data, such as the query results returned by search engines [14] and the online data repositories (e.g. Wikipedia) [2].

For the task of classifying short & sparse text, although limited work has been done in this area, the proposed methods [12][16] similarly resort to external or auxiliary data to deal with the data sparseness problem. For instance, Phan et al. [12] proposed to mine hidden topics from a very large external data collection, and represent each text instance as a vector of words and topics. Then, one classifier was trained on concatenated features from two spaces. The main limitations of the method are: 1) the original features of words and the detected topics are not comparable as they have different physical interpretations and, therefore, different contributions to the classification task. It may not be appropriate to directly concatenating features from different spaces to train a classifier; 2) appending detected topics as new features will increase the dimension of the data, exacerbating the "curse of dimensionality" problem. Hence, in this paper, we aim to construct a classification model for short & sparse text by using external data and dealing with original feature space.

In particular, we propose a method called, **T**ransfer **C**lassification of **S**hort & **S**parse **T**ext (TCSST), which uses transfer learning to exploit the external data to aid the classification for short & sparse text. It is usually assumed by traditional transfer classification that, while there is only a very small set of labeled data in the new target domain, there are plenty of labeled data in the old source domain. Therefore, traditional transfer classification focuses on leveraging the labeled data in the source domain to learn an effective classifier for the target domain. However, for the problem of short & sparse text classification, the external data is typically acquired through querying search engines or crawling online data repositories. It can't be expected that there are plenty of data in the external domain with labels readily available. Therefore, TCSST focuses on exploiting both labeled and unlabeled data in the external domain to construct the classification model for the short & sparse text. We develop novel strategies to allow TCSST to efficiently and iteratively select from the large set of unlabeled external data a subset where the data are very likely to be useful in the transfer classification. The selected data will be used to train the transfer classifier together with other labeled training data.

The main contributions of the work is summarized as follows.

- We proposed a novel short & sparse text classification method which uses transfer learning to exploit the external data and deals with features in the original space.
- We developed novel strategies to leverage both labeled and unlabeled data in the source domain to construct a high quality classification model for the target domain.
- We evaluated the performance of the proposed method on both benchmark and real-world data sets, comparing against baseline and the state-of-the-art approaches.

The rest of this paper is organized as follows. In Section 2, we review existing research on short & sparse text data mining. The problem definition of the transfer classification of short & sparse text data and the proposed method, TCSST, are discussed in details in Section 3. In Section 4, we provide the results of the experiments evaluating the performance of the proposed method. Finally, we summarize the work and discuss the future study in Section 5.

2. RELATED WORK

In this section, we review related research in the areas of short & sparse text mining, from similarity measure to clustering and classification.

Semantic similarity measure between short texts (e.g. words and named entities) has long been an integral part of information retrieval and natural language processing. Existing research can be generally divided into two categories: knowledge-based approaches and corpus-based approaches [10]. The former usually finds the relationship between words using some external lexical database (e.g. WordNet); while the latter frequently identifies the similarity degree by resorting to the information derived from large corpora. For corpus-based approaches, the two representative approaches to obtain large text collection are querying search engines and crawling online data repositories. For example, Bollegala et al. [3] used search engines to obtain enriched context data for measuring similarity between words. Similarly, Sahami and Heilman [14] measured the similarity between short text snippets by leveraging the search engines to acquire greater context. Yin and Meek [18] further improved the work [14] by introducing a web-relevance similarity measure and using a machine learning approach. Gabrilovich and Markovitch [6] computed semantic relatedness by representing the meaning of any text as a vector of Wikipedia concepts.

Clustering short text has attracted growing interests in recent years. Most of the existing research similarly relies on auxiliary data to overcome the data sparseness. For example, Banerjee et al. [2] tried to improve the clustering accuracy for short text by enriching their representation with additional features from Wikipedia. Hu et al. [7] proposed a hierarchical three-level structure to tackle the data sparsity problem and reconstruct the corresponding feature space with the integration of multiple semantic knowledge bases – Wikipedia and WordNet. Recently, Jin et al. [8] proposed to cluster short text messages via transfer learning from auxiliary long text data. While we similarly use transfer learning to exploit the auxiliary data, we focus on a different text mining task – text categorization.

The short and sparse feature pose similar challenges to the task of text categorization. The insufficient word co-occurrence renders traditional techniques such as "Bag-Of-Words" limited. Likewise, existing methods mainly exploit external data to expand the original data. In [16], Sriram et al. proposed to use a small set of features extracted from the author's profile and text to classify *tweets* – microblogs provided by Twitter. Sun et al. [17] used semantic analysis to extract concept information from Wikipedia to deep classify short text. Phan et al. [12] proposed to mine hidden topics from crawled Wikipedia corpus as new features and simply train a classifier from the augmented and combined feature space. However, it may not be appropriate to train a classifier on incomparable features with different physical meanings, and consequently different significance to the classification task. It also results in the increase of data dimensions. In our work, we study a different method – transfer learning – to exploit the external data, such that the classifier is trained in original feature space.

3. THE PROPOSED METHOD

In this section, we first formally define the problem of transfer classification of short & sparse text data. Next, we describe the framework of our proposed method, which is followed by the details of the algorithms.

3.1 Problem Definition

Similar to the existing research in short & sparse text mining, we consider exploiting external data (e.g. Wikipedia) for classifying short & sparse text (details of crawling relevant Wikipedia documents will be discussed in Section 4). Therefore, we have text data from two domains: the short & sparse domain χ^s and the external domain χ^e.

In particular, let $\chi^s = \{x_1^s, x_2^s, \dots, x_m^s\}$ be the set of text

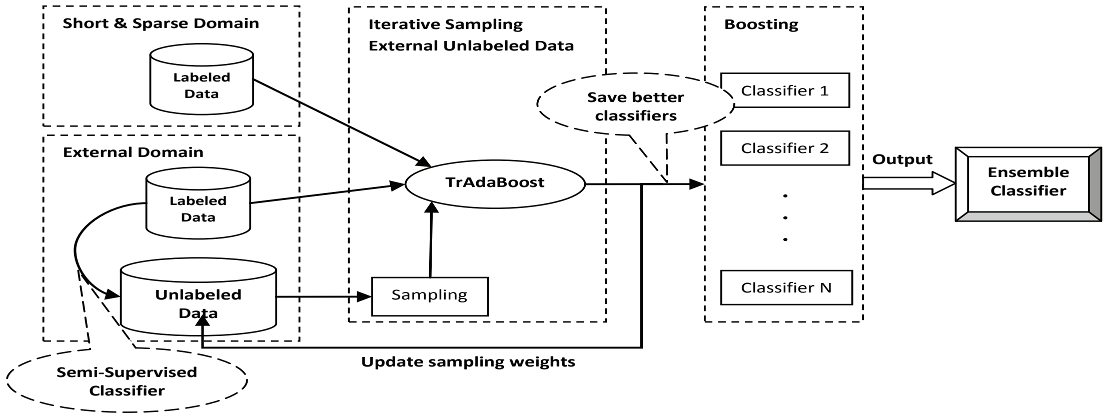


Figure 1: The framework of TCSST.

instances in the short & sparse domain, $\chi^e = \{x_1^e, x_2^e, \ldots, x_n^e\}$ be the set of instances in the external domain, and $\mathcal{C}=\{c_1, c_2, ..., c_k\}$ be the set of category labels. The instances in χ^s are divided into two disjoint subsets: the labeled subset $\mathcal{T}^s = \{(x_i^s, c(x_i^s))\}$ and the unlabeled subset $\mathcal{S}^s = \{(x_i^s)\}$, where $x_i^s \in \chi^s$ and $c(x_i^s) \in \mathcal{C}$. Similarly, χ^e is composed of the labeled subset $\mathcal{T}^e = \{(x_j^e, c(x_j^e))\}$ and the unlabeled subset $\mathcal{S}^e = \{(x_j^e)\}$, where $x_j^e \in \chi^e$ and $c(x_j^e) \in \mathcal{C}$. Then, **the problem of transfer classification of short & sparse text data is**: given a small number of labeled short & sparse text data $\mathcal{T}^s \subset \chi^s$, a small number of labeled external data and a large number of unlabeled external data $\chi^e = \mathcal{T}^e \cup \mathcal{S}^e$, the objective is to learn a classifier $H_{\chi^s \to \mathcal{C}}$ that minimizes the prediction error on the set of unlabeled short & sparse text $\mathcal{S}^s \subset \chi^s$.

There are two major issues that differentiate our problem from traditional transfer classification. Firstly, we focus on dealing with the short & sparse text data. Secondly, we aim to exploit both labeled and unlabeled data in the external domain, while traditional transfer classification leverages only the labeled data in the source domain to help with the classification in the target domain. The reason we try to exploit the unlabeled external data is motivated by the situation in the real-world application. For instance, given a set of labeled short & sparse text, we managed to crawl relevant documents from Wikipedia using carefully selected seeds. However, the labels of the crawled documents may not be readily available, or may be assigned very crudely. In this case, it is only affordable to manually label a very small number of documents in the external domain. Therefore, the proposed method will be practically useful if the unlabeled external data is made good use of to construct a high quality classification model for the short & sparse text data.

3.2 The Framework of TCSST

The framework of our proposed method, Transfer Classification of Short & Sparse Text (TCSST), is shown in Figure 1.

The input data is a small set of labeled short & sparse text. From online data repositories, we should be able to gather a large collection of relevant documents as external data, among which only a small set is labeled. Certainly, without considering the unlabeled external data $\mathcal{S}^e$, existing transfer classification method (e.g. TrAdaBoost [4]) can be employed directly to consider only the labeled data, $\mathcal{T}^s \subset \chi^s$ and $\mathcal{T}^e \subset \chi^e$, to learn a classifier for the short & sparse text. However, due to the insufficient number of labeled external data, the improvement brought by the traditional transfer classification might be limited. Therefore, TCSST aims to further exploit the unlabeled external data.

Intuitively, we can first predict labels for unlabeled external data, using some semi-supervised classification algorithm, in consideration of the limited number of labeled data in the external domain. Then, traditional transfer classification can be applied straightforwardly by treating all external data (with known or predicted labels) as training data.

However, as will be shown in our experiments in Section 4, this straightforward method fails to train a classifier of high quality. Sometimes its performance is even worse than the one using only the labeled external data (referred to as the *base classifier* hereafter). This is because there are too much noise in the unlabeled external data with predicted labels. Therefore, rather than including all unlabeled external instances into the transfer classification, we devise an iterative sampling process to allow TCSST to select a small set of unlabeled external data in each round. The selected unlabeled external data, with predicted labels, will be used together with other labeled data to train a transfer classifier. Each unlabeled external instance is associated with sampling probability, which will be updated in each round to reflect the importance of the instance to the transfer classification. Subsection 3.6 describes the details of the sampling process. Basically, an unlabeled external instance will be more likely to be sampled if it is regarded to be more useful in the transfer classification. If the learned transfer classifier works better than the base classifier, the classifier will be saved. As shown in the Figure 1, the output of TCSST is an ensemble classifier consisting of all intermediate classifiers better than the base classifier.

The main steps of TCSST are outlined as follows.

1. Predict labels for unlabeled external data (Subsection 3.4);
2. Assign initial sampling probabilities for unlabeled external data (Subsection 3.6) ;
3. Sample a set of unlabeled external data to train a

transfer classifier (Subsection 3.5), together with labeled external data and labeled short & sparse data;

4. Save or discard the transfer classifier depending on its performance and update the sampling probabilities for unlabeled external data (Subsection 3.6);

5. Go back to step 3 or output the boosting classifier on saved classifiers (Subsection 3.7).

3.3 Data Preparation

Today, Wikipedia has been known as the richest online encyclopedia which includes a huge number of documents in various topics. Since Wikipedia covers a lot of concepts and domains, it is reasonable to extract parts of Wikipedia documents to form an external dataset related to the target domain.

To collect data from Wikipedia, we prepare various seed crawling keywords from different class domains. The simplest method is utilizing the class name as the seed keywords. For example, in our five-class dataset, we utilize five class names (Human, Natural, Social, Physical, Financial) as crawling keywords. Furthermore, if the class labels are inconsistent or irrelevant to instances context, we can generate some keywords from the context information by common sense. For example, Phan et al. [12] generated seventy keywords for classification task. We crawl the seed's corresponding Wikipedia page and follow the outgoing hyperlinks to acquire related pages. Each crawling transaction is limited by the maximum depth of hyperlink (usually 4).

After downloading the pages, we would like to assign labels for these pages. Intuitively, for each seed, we can assign class label by manual, and assign the same label to the seed' related pages. However, this method has two limitations. Firstly, different seeds may share same related pages. Secondly, as the hyperlink depth increasing, the content of related pages are surprisingly different to the original seed page. Consider the limitations above, we manually assign labels for part of documents, and leave others as unlabeled data.

3.4 Semi-supervised Classification

In order to use the unlabeled external data in the transfer classification framework, we first predict labels for unlabeled external data. Considering the limited number of labeled external data, we choose to use semi-supervised classification instead of supervised classification. In particular, we use the Naive Bayes based self-training approach [19], while any semi-supervised classification algorithm should be workable here. The basic idea of self-training is to train firstly a classifier with the small amount of labeled data. The classifier is then used to classify the unlabeled data. Typically the most confident unlabeled data, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure repeated.

To avoid the over-fitting problem, we add a bagging framework on classifiers trained by the self-training at each iteration, and output an ensemble classifier $H(x)$ as the final classifier of semi-supervised learning.

$$H(x) = arg\min_{c_k \in \mathcal{C}} \sum_i (|h_i(x) - c_k|) \tag{1}$$

where $x \in \mathcal{S}^e \subset \chi^e$, and h_i is the classifier trained at the i^{th} iteration of the semi-supervised learning.

3.5 Transfer Classification

According to the categorization of transfer learning in [11], our task of classifying short & sparse text using external data belongs to the category of *inductive transfer learning*. Different types of inductive transfer learning methods have been proposed, such as the feature representation based transferring [13] and the parameter based transferring [5]. In our work we adopt the *instance based transferring*, which provides a more flexible framework to allow us to include unlabeled external data instances afterward.

In particular, we borrow the framework of TrAdaBoost [4], which is a representative instance based transferring classification method. It extends from AdaBoost [15] that aims to boost the accuracy of a weak learner by carefully adjusting the weights of training instances and learn a classifier accordingly. Similar to AdaBoost, TrAdaBoost iteratively increases the weights for instances from the target domain (e.g. short & sparse domain) if they are wrongly predicted, expecting that the subsequent classifiers built will be tweaked in favor of the instances misclassified by previous classifiers. For instances in the source domain (e.g. external domain), the weights of wrongly predicted instances will be decreased, because it is regarded that these instances are most dissimilar to those in the target domain, exerting less impact on the classification.

In particular, the weight of instances in the target domain is updated as follows.

$$w_i^{(t+1)} = w^{(t)} * (\frac{1-\epsilon_t}{\epsilon_t})^{|h_t(x_i)-c(x_i)|}, \epsilon < 1/2 \tag{2}$$

where x_i is the i^{th} labeled instance in the target domain, ϵ_t is the t^{th} classifier's average lost over the target domain, and $|h_t(x_i) - c(x_i)|$ represents the t^{th} classifier's lost on the instance x_i.

For wrongly predicted instances in the source domain, the weight is decreased based on the maximal iteration number, N, and the size of the data set n.

$$w_i^{(t+1)} = w^{(t)} (\frac{1}{1+\sqrt{2 \ln n/N}})^{|h_t(x_i)-c(x_i)|} \tag{3}$$

Classifiers trained from iterations 0 to $N/2$ are supposed to filter dissimilar instances in the source domain. Hence, TrAdaBoost finally outputs an ensemble classifier which is a weighted combination of the classifiers trained from iteration $N/2$ to N.

$$H(x) = arg\min_{c_k \in \mathcal{C}} \sum_{t=\lceil N/2 \rceil}^{N} ((\log \frac{1-\epsilon_t}{\epsilon_t}) * |h_t(x) - c_k|) \tag{4}$$

where $\log \frac{1-\epsilon_t}{\epsilon_t}$ is the weight of the classifier h_t.

3.6 Sampling Unlabeled External Data

Recall that, due to the small number of labeled external data, TCSST aims to exploit the large set of unlabeled external data to boost the transfer classification. In particular, TCSST first predicts labels for unlabeled external data and then iteratively selects a small set of unlabeled external data, with predicted labels, to learn the transfer classifier together with the other training data (i.e. the labeled external data and labeled short & sparse data). In this subsection, we describe the strategies TCSST uses to sample unlabeled external data.

The basic idea is that, the more likely an instance is useful in the transfer classification, the more possible it will be sampled. To implement the idea, we assign a *sampling probability* to each unlabeled instance in the external domain. Given an instance $x_i \in \mathcal{S}^e \subset \chi^e$, an initial sampling probability $\rho(x_i)^0$ will be assigned. We first describe how to update $\rho(x_i)^{t+1}$ based on $\rho(x_i)^t$. We will revisit the point to discuss how to assign the initial sampling probability $\rho(x_i)^0$ later.

As introduced in Subsection 3.5, TrAdaBoost updates the weights for training instances in the external domain after each iteration. In particular, the weight of an instance will be decreased if it is regarded to be less useful in the transfer classification (i.e. Eq. 3). We will use the by-product of TrAdaBoost to update the sampling probability for unlabeled external data. First, a set of unlabeled external instances $\mathcal{P}^0 = \{x_1, x_2, \ldots, x_p\} \subset \mathcal{S}^e$ is selected, based on some initial sampling probabilities, to be included in TrAdaBoost. Each instance x_i will be assigned a sequence of weights respectively in the N iterations of TrAdaBoost $< w_i^{(0)}, w_i^{(1)}, \ldots, w_i^{(N)} >$. We define the final weight of the instance, ω_i, as the sum of the weights assigned in the iterations from $\lceil N/2 \rceil$ to N where the classifiers start to converge.

$$\omega_i = \sum_{t=\lceil N/2 \rceil}^{N} w_i^{(t)} \tag{5}$$

Let the final weights of the instances in $\mathcal{P}^0$ be $< \omega_1, \omega_2, \ldots, \omega_p >$. We normalize the weights by setting ω_i to $\omega_i / \sum_{j=1}^{p} \omega_j$ such that $\omega_i \in (0, 1)$. Then, we update the sampling probability for each unlabeled external instance as follows.

$$\rho(x_i)^{(t+1)} = \begin{cases} \omega_i * \rho(x_i)^{(t)}, & \text{if } x_i \in \mathcal{P}^{(t)} \\ \rho(x_i)^{(t)}, & \text{otherwise} \end{cases} \tag{6}$$

For an unlabeled instance that has been selected in the t^{th} round, if it turns out to be useful in the transfer classification (e.g. the weight ω_i is high), Eq. 6 will decrease (because $\omega_i < 1$) its sampling probability less significantly. Otherwise, its weight ω_i will be low and its sampling probability will be decreased more significantly. If an unlabeled instance has not been selected in the t-th round, its sampling probability will not be decreased. That is, an unsampled instance will have more chances to be selected.

We now discuss how to assign the initial sampling probability $\rho(x_i)^{(0)}$ for each unlabeled external data. Since we have a large collection of unlabeled external data, if starting from a set $\mathcal{P}^0$ consisting of instances that are not useful in the transfer classification, it may require more rounds of iterative sampling to find useful data to boost the classification. Therefore, instead of assigning equal sampling probabilities to all unlabeled external data, we assign higher sampling probabilities to instances which are more likely to be useful. In particular, we learn a base classifier by running TrAdaBoost on labeled short & sparse text (e.g. $\mathcal{T}^s$) and labeled external text (e.g. $\mathcal{T}^e$) only. Based on the results of the base classifier, we can similarly compute the final weight for each labeled external instance, $\omega(x_j)$, using Eq. 5 and normalize to make sure $\omega(x_j)$ ranges between 0 and 1. Next, given an unlabeled external instance $x_i \in \mathcal{S}^e$, we measure its similarity with every labeled external instance $x_j \in \mathcal{T}^e$. If it is more similar to a high quality labeled external instance, its

Algorithm 1 Transfer Classification of Short & Sparse Text

```
input
  $\mathcal{T}^s, \mathcal{T}^e, \mathcal{S}^e$, $M$ - size of $\mathcal{P}$, MAX_ITERATION
output
  $H(x)$ - the TCSST classifier
 1: TrAdaBoost($\mathcal{T}^s$, $\mathcal{T}^e$) /*learn the base classifier $h_0(x)$*/
 2: For each $x_i \in \mathcal{S}^e$, assign $\rho(x_i)^{(0)}$ as Eq. 7
 3: Let $t = 0$, $H = \emptyset$
 4: while t<MAX_ITERATION do
 5:    Sample $\mathcal{S}^e$ to generate $\mathcal{P}^{(t)}$
 6:    $h_t(x)$ =TrAdaBoost($\mathcal{T}^s$, $\mathcal{T}^e$, $\mathcal{P}^{(t)}$)
 7:    if $\alpha_t > \alpha_0$ then
 8:       $H = H \cup \{h_t(x)\}$ /*$\alpha_i$ is $n$-fold cross-validation
          accuracy of $h_i$*/
 9:    end if
10:    t= t+1
11:    For each $x_i \in \mathcal{S}^e$, update $\rho(x_i)^{(t)}$ as Eq. 6
12: end while
13: $H(x) = arg\min_{c_k \in \mathcal{C}} \sum_{i=1}^{|H|} (\alpha_i * |h_i(x) - c_k|)$
```

initial sampling probability will be higher. That is,

$$\rho(x_i)^{(0)} = \sum_{x_j \in \mathcal{T}^e} sim(x_i, x_j) * \omega(x_j) \tag{7}$$

where $sim(x_i, x_j)$ is the similarity between two text instances and the cosine similarity measure is adopted here.

3.7 Boosting From Saved Classifiers

As aforementioned, TCSST iteratively samples a set of unlabeled external instances with predicted labels, together other labeled training data to learn a transfer classifier. If the learned classifier performs better than the base classifier which uses labeled training data only, the learned classifier will be saved. The final output of TCSST is an ensemble classifier combining all saved classifiers. We use the n-fold cross-validation accuracy over the training data as the voting weight of each saved classifier $h_i(x)$. Then, the output classifier $H(x)$ of TCSST is,

$$H(x) = arg\min_{c_k \in \mathcal{C}} \sum_i (\alpha_i * |h_i(x) - c_k|) \tag{8}$$

where α_i is the n-fold cross validation accuracy of the i-th saved classifier.

The pseudo code of TCSST is given in Alg. 1. Firstly, we learn a base classifier by training TrAdaBoost on labeled training data (line 1), based on which we assign initial sampling probabilities to unlabeled external data (line 2). Then, an iterative process is started by sampling a subset of unlabeled external data $\mathcal{P}^{(t)}$ (line 5), which will be used to train the t-th transfer classifier $h_t(x)$ (line 6). If the performance of the t-th classifier is better than the base classifier $h_0(t)$, $h_t(x)$ will be saved (lines 7-9). Based on the results of t-th classifier, the sampling probabilities of unlabeled external instances are updated (line 11). The final classifier is a weighted combination of saved classifiers (line 13).

4. PERFORMANCE STUDY

In this section, we evaluate the performance of TCSST. We describe first the data sets used in our experiments. Then, the methods used to be compared with TCSST will

Table 1: Data set constructed from 20-Newsgroup

Data Set	Short & Sparse D.	External D.
rec vs talk	rec.autos	rec.sport.baseball
	rec.motorcycles	rec.sport.hockey
	talk.politics.guns	talk.politics.mideast
	talk.politics.misc	talk.religion.misc

Table 2: Statistics of the two data sets

	Benchmark Data	NRM Data
Size of χ^s	3669	189
Size of χ^e	3561	1137
Sparse degree	99.93% $\longrightarrow$ 99.98%	99.92%
Vocabulary size $\|V\|$	15254	6112
Number of classes	2	5

be discussed. Finally, we show the results of different types experiments.

4.1 Datasets

We evaluate the performance of TCSST on a benchmark data set as well as a real-world data set.

For the benchmark data set, we consider the well-known 20newsgroup data [9], which has been frequently used for evaluating and comparing different transfer learning algorithms for text classification. The 20newsgroup is a text collection of approximately $20,000$ news-group documents, which are partitioned across 20 different newsgroups nearly evenly. In our study, we generate a data set from two top categories, *rec* and *talk*, one as positive and the other as negative. We then split the data in the two categories based on their sub-categories. Table 1 shows the specific sub-category based splitting strategy that divides the data into two domains. The splitting strategy ensures that the documents in the two domains are different but related, since they are under different sub-categories while same top categories. The task is to classify top categories. Since we focus on dealing with the classification of short & sparse text, we make one of the domains (the one in the second column of Table 1) to be sparse, by randomly deleting features for instances. The other domain (i.e. the third column of Table 1) then serves as the external domain. In our experiments, we will vary the parameter $-$ the percentage of features deleted from each instance $-$ to generate sparse domains with different sparse degrees, and evaluate classification models on the generated sparse data sets. The second column of Table 2 summarizes some statistics of the benchmark data set. The top two numbers are the sizes of the short & sparse domain and the external domain respectively. The third number is the sparse degree of the short & sparse domain, which is defined as follows. Let M be a matrix representation of the short & sparse domain χ^s such that m_{ij} is non-zero if the i-th instance contains the j-th word. Then, the sparse degree of χ^s is defined as the ratio of zero elements to all elements in M.

$$Sparse\_degree(\chi^s) = \frac{|\{m_{ij}|m_{ij}=0\}|}{|\chi^s| * |V|} \qquad (9)$$

where $|V|$ is the vocabulary size. Thus, the higher the sparse degree, the more sparse the data.

We also use a real-world data set to evaluate the classification models. The data was collected from participatory workshops that were held to assess the capacity of natural resource managers, such as landholders and graziers, to adapt to natural resource changes. Participants were invited to comment from five indicators: *human*, *social*, *natural*, *physical* and *financial*. Each comment thus is a short & sparse text instance, belonging to one of the five categories. To obtain relevant external data, we follow the strategy used in [12] to crawl Wikipedia documents. We use the five class labels as the seed keywords. For each seed keyword, we ran JWikiDocs [1] to download the corresponding Wikipedia pages. We also crawled the relevant pages by following outgoing hyperlinks, with the maximal depth of hyperlink as 3. We crawled overall 1137 relevant documents from Wikipedia. After removing HTML tags, noisy text and links, stop words, the statistics of the real-world data set are reported in the third column of Table 2. We refer to this data set as NRM Data and make it available online [1].

4.2 Compared Methods

We compare the performance of TCSST against the following methods.

- Supervised Classification. This method uses only the small set of labeled short & sparse text $\mathcal{T}^s \subset \chi^s$ as the training data to learn a classifier. It serves as the baseline approach. In particular, we use Random Tree as the classification model and refer to this method as *Random Tree.*

- Traditional Transfer Classification. This method uses only the small set of labeled short & sparse text $\mathcal{T}^s \subset \chi^s$, and the small set of labeled external data $\mathcal{T}^e \subset \chi^e$ as the training data to learn a transfer classifier. In particular, we use TrAdaBoost as the classification model, and RandomTree as the weak learner used by TrAdaBoost. Note that, TCSST is an ensemble classifier which combines multiple TrAdaBoost classifiers. To enable a fair comparison, we randomly generate the same number of TrAdaBoost classifiers using $\mathcal{T}^s \cup \mathcal{T}^e$ only, and then boost from the set of TrAdaBoost classifiers. We refer to the algorithm as *Ensemble-TrAdaBoost.*

- Feature Augmented Classification. This is the state-of-the-art method proposed in [12], which augments the feature vector with topics detected from large collection of external data. We refer to this method as *LDA-Classification.*

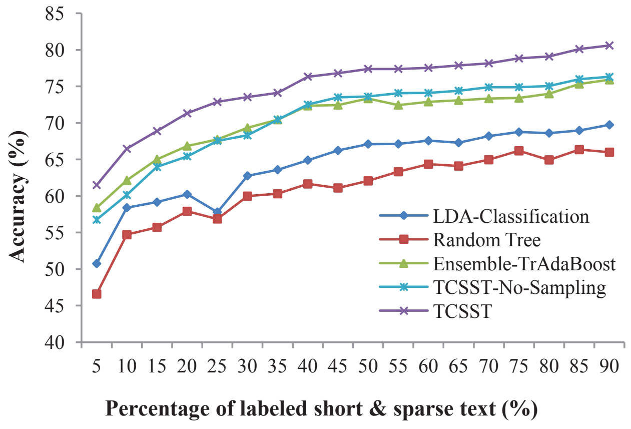
- TCSST without sampling. This is the method which simultaneously includes all of the unlabeled external data $\mathcal{S}^e \subset \chi^e$, together with other labeled data $\mathcal{T}^s \cup \mathcal{T}^e$, to train the TrAdaBoost classifier. Similarly, the weak learner used by TrAdaBoost is RandomTree. We refer to method as *TCSST-No-Sampling.*

Similar to Ensemble-TrAdaBoost and TCSST-No-Sampling, we consistently use RandomTree as the weak learner of TrAdaBoost for TCSST.

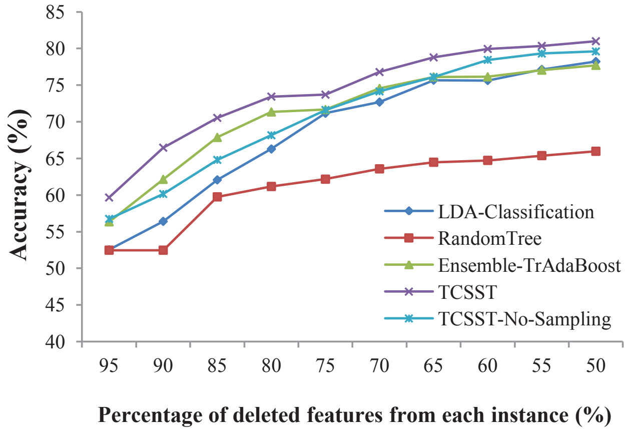
[1]JWikiDocs: http://jwebpro.sourceforge.net

Table 3: Accuracy of TCSST and other methods on the benchmark data

Percentage of removed features	Classifier	5% labeled short sparse data	10% labeled short sparse data	15% labeled short sparse data	20% labeled short sparse data
90 %	Random Tree	53.9171 %	56.7847 %	57.9801 %	60.3508 %
	Ensemble-TrAdaBoost	58.4065 %	62.1435 %	66.8512 %	66.8512 %
	LDA-Classification	50.7497 %	58.4066 %	59.1630 %	60.2287 %
	TCSST-No-Sampling	56.7604 %	60.1473 %	63.9788 %	66.4339 %
	TCSST	**61.5127** %	**66.4648** %	**68.9003** %	**71.3215** %
95 %	Random Tree	51.6638 %	52.4735 %	54.4758 %	56.3420 %
	Ensemble-TrAdaBoost	54.4062 %	56.3427 %	58.4514 %	59.9336 %
	LDA-Classification	46.4033 %	55.5250 %	55.5675 %	57.8508 %
	TCSST-No-Sampling	**56.1102** %	56.7464 %	58.5123 %	59.2643 %
	TCSST	55.9811 %	**59.6654** %	**60.3719** %	**63.3174** %



(a) Variation of the size of labeled short & sparse text



(b) Variation of the sparse degree

Figure 2: Comparison of TCSST and other methods along a wide range of data variation.

4.3 Experimental Results

4.3.1 Comparison between algorithms.

Firstly, we compare the performance of TCSST against the other methods on the benchmark data set. We vary the parameter, percentage of features deleted from instances, from 90% to 95% to generate two sparse domains, while the external domain is not changed. The results of the classifiers mentioned in Subsection 4.2 as well as those of TCSST on the two data sets are shown in Table 3. In this set of experiments, we respectively set the percentage of labeled data in the short & sparse domain as 5%, 10%, 15% and 20%. For Ensemble-TrAdaBoost, TCSST-No-Sampling and TCSST, the percentage of the labeled external data is set to 10%. For TCSST, in each round, 20% of unlabeled external data is selected to be included in the training data.

For Ensemble-TrAdaBoost, TCSST-No-Sampling and TCSST, the number of iterations used by TrAdaBoost is set to 40. For Ensemble-TrAdaBoost and TCSST, the number of TrAdaBoost classifiers is set to 20. For LDA-Classification, 100 topics are detected from the sparse domain and the external domain.

All results listed in Table 3 are generated from 10-fold cross validation. It can be observed that TCSST outperforms all of the other four algorithms in nearly every setting. Comparing Random Tree and Ensemble-TrAdaBoost, we notice that transfer learning using external data improves the classification using labeled sparse data only. Comparing Ensemble-TrAdaBoost and LDA-Classification, it shows that transfer classification which deals features in the original space is better than the existing approach which trains one classifier by concatenating features from different spaces. For TCSST-No-Sampling, we observe that its performance is not stable. It works better than Ensemble-TrAdaBoost sometimes (and better than TCSST once), and worse than Ensemble-TrAdaBoost some other times. This might be caused by the noise in the large set of unlabeled external data. Therefore, the experimental results demonstrate the superiority of TCSST.

While Table 3 lists the accuracy results of the classification algorithms at a few settings of the two parameters: the percentage of labeled short & sparse text and the percentage of features deleted from each instance, Figure 2 plots the performance trend of each method along a wide range variations of the two parameters. Figure 2 (a) shows the results when the percentage of labeled short & sparse text is varied from 5% to 90% while the percentage of features deleted is set to 90%. It can be observed that TCSST clearly and consistently outperforms other methods along the whole variation range. Figure 2 (b) shows the results when the percentage of features deleted from each instance is varied from 95% to 50%, while the percentage of labeled short & sparse text data is 10%. Again, TCSST improves over all of the other four algorithms, while the improvement margin decreases when the data becomes less sparse.

Table 4: Accuracy of TCSST and other methods on the NRM data

Classifier	50% labeled short sparse data
Random Tree	32.8042 %
Ensemble-TrAdaBoost	42.1053 %
LDA-Classification	46.0317 %
TCSST-No-Sampling	44.5089 %
TCSST	**50.8772** %

The results of the algorithms on the NRM data is shown in Table 4. For Ensemble-TrAdaBoost, TCSST-No-Sampling and TCSST, the number of labeled data in the external domain is set to be the same as the number of labeled data in the short & sparse domain (i.e. $50\% * 189 = 95$). For TCSST, in each round, twice of the number of labeled data in the external domain is selected to be included as training data (i.e. $95 * 2 = 190$). The other settings of each classifier are same as in the previous experiment. Again, TCSST achieves the best performance on the real-world data.

4.3.2 *Parameter Influence on TCSST*

We also conduct experiments to evaluate the influence of parameters on the performance of TCSST. In particular, we vary the two parameters: the percentage of labeled external data and the size of the unlabeled external data selected in each round. The results are shown in Figures 3 (a) and (b) respectively. The experiments were run on the benchmark data set where the sparse domain is generated by deleting 90% of features from instances. For both experiments, the percentage labeled short & sparse text data is 20%.

For Figure 3 (a), we vary the percentage of labeled external data from 5% to 50% and set the size of unlabeled external data selected in each round to be 20%. It can be observed the performance of TCSST will be better given more labeled external data. The results can be understood intuitively.

For Figure 3 (b), we vary the size of the unlabeled external data selected in each round from 5% to 50%, and set the percentage labeled external data to be 10%. We notice that, when the parameter increases, that is, more unlabeled external data is selected in each round, the performance of TCSST will decrease slightly. This is probably because when more unlabeled external data is selected in each round, some noise is introduced.

4.3.3 *Iterations vs Accuracy of TCSST*

We further carry out experiments by varying the number of MAX_ITERATION in the Algorithm 1. That is, the number of iterative sampling process. We are interested in finding out how many iterations it takes before TCSST's accuracy becomes stable.

We run the experiment on the benchmark data set where the sparse domain is generated by deleting 90% of features from each instance. The percentage of labeled short & sparse text is 20%, the percentage of labeled external data is 10% and the size of unlabeled external data selected in each round is 20%. The experimental results are shown in Figure 4. It can be observed from the figure that TCSST becomes stable approximately when the MAX_ITERATION reaches 20. Given the large set of unlabeled external data, we believe the result is quite promising. It indicates that our strategies of starting with a good initial set $\mathcal{P}^{(0)}$ of unlabeled external data, and updating sampling probabilities based on the usefulness of unlabeled instances are effective.

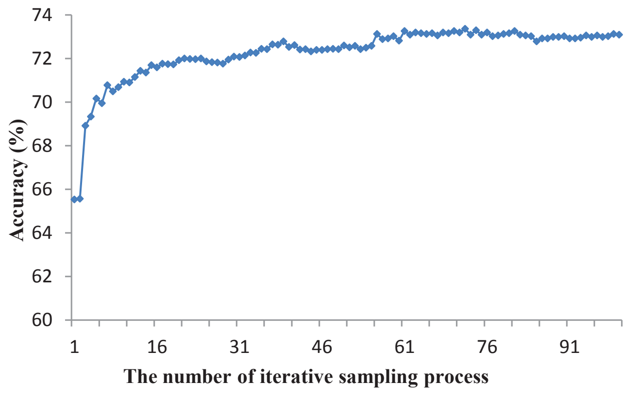


Figure 4: Iterations vs Accuracy of TCSST w.r.t. MAX_ITERATION.

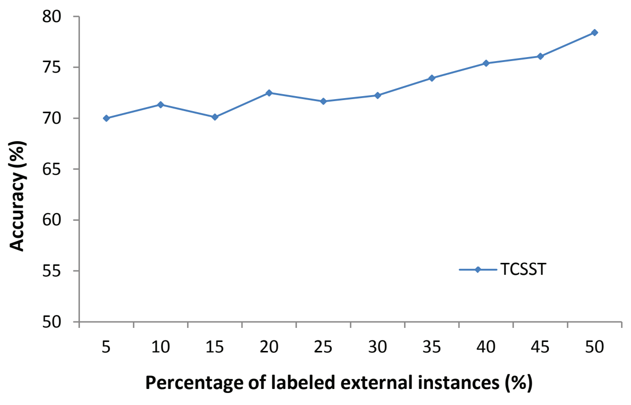
4.3.4 *Computational Complexity of TCSST*

TCCST's computational performance is related to three factors: C - weak classifier's complexity, N - number of iteration for training classifier in TrAdaBoost, and M - number of iteration for sampling in TCCST. The computational complexity of TCSST is $O(MNC)$.

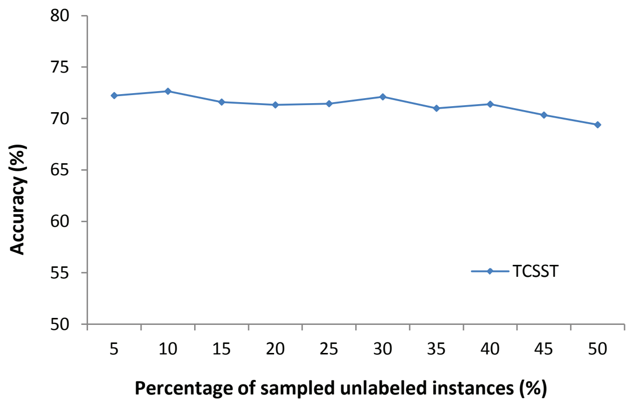
Accurately the three factors above are not independent. For example, the weak classifier's complexity C is usually related to the training data scale which influences the selection of TrAdaBoost iteration number N. Furthermore, the N decides the external data sampling size (S) which empirically choose $10\% - 20\%$ of N. Obviously, the S acts as the learning rate to induce the TCCST's sampling iteration M.

5. CONCLUSIONS & FUTURE WORK

In this paper, we proposed a novel method, TCSST, which classifies short & sparse text by using transfer learning to exploit the external data. Motivated by the situation that there may not be enough labeled external data, TCSST leverages both labeled and unlabeled external data for transfer classification. In order to avoid the noise in the large set of unlabeled external data, TCSST uses an iterative sampling process to include only a subset of high quality unlabeled external data, with predicted labels, in each round into the training data, rather than including all unlabeled external instances into the training data in a batch. The experimental results demonstrate the effectiveness of the iterative sampling process and the strategies to allow the selection of



(a) Variation of the size of labeled external instances



(b) Variation of the size of sampled unlabeled external instances

Figure 3: Parameter influence on TCSST.

external instances that are more likely to be useful. Compared with baseline as well as the existing methods, TCSST presents superior performance across benchmark and real-world data sets.

While TCSST borrows the framework of TrAdaBoost, which focuses on transferring the knowledge of instances, there are other types of inductive transfer learning, such as transferring knowledge of feature representations, transferring knowledge of parameters, and transferring relational knowledge [11]. As ongoing research, we will study how to employ other types of inductive transfer learning frameworks to classify short & sparse text using a small set of labeled external data and a large set of unlabeled external data, or using only a large set of unlabeled external data.

6. ACKNOWLEDGMENTS

This work was supported, in part, by the Australian Research Council (ARC) Linkage Project under Grant No. LP120100566, Discovery Project under Grants No. DP1093762, and the NSW Department of Environment, Climate Change and Water (DECCW).

7. REFERENCES

[1] Natural resource manager capacity accessment data. In *http://sourceforge.net/projects/tcsst/*.

[2] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using wikipedia. In *Proc. ACM SIGIR*, 2007.

[3] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *Proc. WWW*, 2007.

[4] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proc. ICML*, 2007.

[5] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proc. KDD*, 2004.

[6] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. IJCAI*, 2007.

[7] X. Hu, N. Sun, C. Zhang, and T.-S. Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proc. CIKM*, 2009.

[8] O. Jin, N. N. Liu, K. Zhao, Y. Yu, and Q. Yang. Transferring topical knowledge from auxiliary long texts for short text clustering. In *Proc. CIKM*, 2011.

[9] K. Lang. Newsweeder: Learning to filter netnews. In *Proc. ICML*, 1995.

[10] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proc. AAAI*, 2006.

[11] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359, 2010.

[12] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proc. WWW*, 2008.

[13] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proc. ICML*, 2007.

[14] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. WWW*, 2006.

[15] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explnation for the effectiveness of voting methods. In *Proc. ICML*, 1997.

[16] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *Proc. SIGIR*, 2010.

[17] X. Sun, H. Wang, and Y. Yu. Towards effective short text deep classification. In *Proc. SIGIR*, 2011.

[18] W. Yih and C. Meek. Improving similarity measures for short segments of text. In *Proc. AAAI*, 2007.

[19] X. Zhu. Semi-supervised learning literature survey. In *http://pages.cs.wisc.edu/˜ jerryzhu/pub/ssl_survey.pdf*, 2008.