CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2016; **28**:3691–3706
Published online 27 May 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3879

Building text classifiers using positive, unlabeled and 'outdated' examples

Jiayu Han[1], Wanli Zuo[1,2], Lu Liu[1], Yuanbo Xu[1] and Tao Peng[1,2,*,†]

[1]*College of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2]*Key Laboratory of Symbol Computation and Knowledge Engineering Ministry of Education, Changchun 130012, China*

SUMMARY

Learning from positive and unlabeled examples (PU learning) is a partially supervised classification that is frequently used in Web and text retrieval system. The merit of PU learning is that it can get good performance with less manual work. Motivated by transfer learning, this paper presents a novel method that transfers the 'outdated data' into the process of PU learning. We first propose a way to measure the strength of the features and select the strong features and the weak features according to the strength of the features. Then, we extract the reliable negative examples and the candidate negative examples using the strong and the weak features (Transfer-1DNF). Finally, we construct a classifier called weighted voting iterative support vector machine (SVM) that is made up of several subclassifiers by applying SVM iteratively, and each subclassifier is assigned a weight in each iteration. We conduct the experiments on two datasets: 20 Newsgroups and Reuters-21578, and compare our method with three baseline algorithms: positive example-based learning, weighted voting classifier and SVM. The results show that our proposed method Transfer-1DNF can extract more reliable negative examples with lower error rates, and our classifier outperforms the baseline algorithms. Copyright © 2016 John Wiley & Sons, Ltd.

Received 19 December 2015; Revised 28 April 2016; Accepted 30 April 2016

KEY WORDS: text classification; strong features; weak features; Transfer-1DNF; WV-ISVM

1. INTRODUCTION

With the rapid development of the Internet and E-commerce, a substantial portion of data exists in the Web. The text data are the commonest expression forms of the information. Text classification is an important and useful technique in the field of data mining and web mining. Many machine learning algorithms are used in text classification [1]. Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels [2].

There are two types of text classification: the supervised classification and the partially supervised classification. The supervised classification needs many manual labeling, which is very labor-intensive and time-consuming. Therefore, the problem we study is how to build the classifiers using the positive examples (identified by *P*) and the unlabeled examples (identified by *U*). This problem is called PU learning, and it is a task of the partially supervised classification. Compared with the supervised learning, PU learning only needs labeled positive examples that reduce much of manual work. PU learning is an essentially binary classification, but the techniques used in the supervised classification cannot be applied in the PU learning directly.

*Correspondence to: Tao Peng, College of Computer Science and Technology, Jilin University, Changchun 130012, China.
†E-mail: tpeng@jlu.edu.cn

There are several typical approaches are proposed in the literature to address PU learning problem, such as spy EM (S-EM) [3], positive example-based learning (PEBL) [4], one-class support vector machines (SVMs) [5], Roc-SVM[6], weighted logistic regression [7] and based SVM [8]. PU learning is widely applied in Web and text retrieval applications, such as finding positive documents from a large number of document collections [3], spotting fake reviews [9, 10] and so on. Most of the time, we only pay attention to a small part of information instead of all the information in the Web. For example, the sports enthusiasts are interested in the areas about sport games, sports stars and so on. Generally, the category that the users interested in is called positive category, and the others are called negative category.

Building a classifier needs two kind of labeled examples: positive examples and negative examples. However, PU learning does not need the negative examples that are labeled manually. In general, PU learning contains two steps: selecting the reliable negative examples (*RN* for short) from *U* and building the classifier using the *P* and the *RN*. The reliable negative examples are negative examples, and they are called 'reliable' because although they are not labeled manually, they are with high reliability. There are two kinds of methods to select the reliable negative examples. One is based on the examples themselves. This kind of method needs to build some classifiers and uses the classifiers to select a set of *RN* from *U*, such as naïve Bayesian technique [8] and Rocchio techniques [6]. The other one is based on the feature distribution of the examples, such as 1DNF. This kind of method first builds a positive feature set *PF* according to the feature frequency, respectively, in P and in U, and the example is regarded as a reliable negative one if it does not have any features in *PF*. We compare the two kinds of methods from three aspects. From the aspect of the selecting efficiency, the latter method is more efficient than the former method because the former method needs to run multiple times to build classifiers. From the aspect of the quality of the negative examples, the latter method has lower error rates than the former method because the latter one has a strict constraint condition. From the aspect of the quantity of negative examples, the former method can get more examples than the latter method because the classifiers can select more examples. There are also two types of methods in the second step. One is running a classification algorithm using *P* and *RN*. This method works well if *RN* is sufficiently large and contains mostly negative documents [11]. When *RN* is small, we should build the classifier using *P*, *RN* and *U-RN* by running an existed algorithm iteratively, and the learning process can stop until it meets some stopping criterion.

Because there is no labeled negative example in the training set of PU learning, the efficiency and the effectiveness of PU learning depend on the quality and the quantity of the reliable negative examples. Therefore, the first step is the basic and the key part to PU learning. Throughout our analysis, we found that there is an inverse relationship between the number of the positive features and the number of the reliable negative examples. As is shown in Figure 1, the number of the reliable negative examples is decreasing with the increasing number of the positive features.

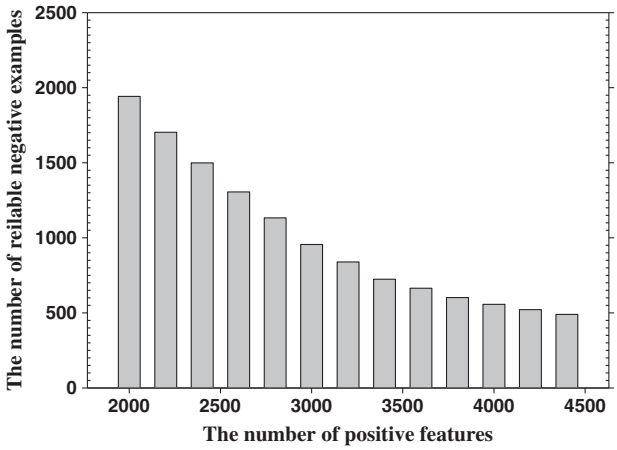

Figure 1. The relationship between the number of positive features and the number of negative examples.

DOI: 10.1002/cpe

In this paper, we present a novel method Transfer-1DNF that is based on 1-DNF [4] algorithm for the first step. The shortcoming of the 1-DNF is that it cannot extract enough reliable negative examples, because it uses lots of positive features. In general, the examples in the training set can be organized in a hierarchical structure. By analyzing the relations between the examples in different categories, we find that the examples from different categories that are under the same top category share more features. Inspired by the idea of transfer learning [12], we use the outdated data to help us to address this shortcoming. The outdated data is the data that is obtained in one time period and may not follow the same distribution in a later time period. We first give a calculation method to measure the strength of the feature relative to the positive category and further choose the strong and the weak features of the positive category according to the feature strength. We use the strong features and weak features to select the reliable negative examples and the candidate negative examples. The candidate negative examples are used to supplement the reliable negative examples when necessary. Next, we build the classifiers with the positive examples, the reliable negative examples and the unlabeled examples by applying SVM iteratively. At each iteration, we measure the subclassifier using a validation set and assign the subclassifier a weight. Finally, we get more than one classifier, and then, we use them to classify the examples by weighted voting method. The experiments show that our method can extract abundant reliable negative examples efficiently, and the classifier we build can get a better performance than the existing state-of-the-art PU classification methods.

The rest of this paper is organized as the following. We first briefly review PU learning in Section 2. Then, Section 3 introduces the techniques we use to select the negative examples in details, and Section 4 describes how to build classifiers. In Section 5, we report experimental results to demonstrate the effectiveness and efficiency of our proposed methods. Finally, we give a brief conclusion about our work in Section 6.

2. RELATED WORK

In the early stage research, there are several research [13–16] that only used labeled positive data and discarded the unlabeled data. Much of valuable knowledge that can be used in the unlabeled data were ignored in these methods; therefore, their performance are poorer than the learning methods such as PU learning that can take full advantage of the unlabeled data.

PU learning is a kind of semi-supervised learning problem based on constrained optimization idea. The key feature of PU learning is that it only needs the labeled positive examples and the unlabeled examples for training, so the training data is abundant and is easy to obtain. A great deal of works has focused on the PU learning because they are widely used in many applications. Over the years, there are many practical algorithms proposed. S-EM [3], PEBL [4], Rocchio [6] and biased-SVM [8] are the typical learning models, and they have a great significance to the following works.

There are two kinds of approaches for PU learning in the literatures: two-step approach and direct approach. In the two-step approach, the first step is to extract the reliable negative examples from the unlabeled dataset. There are many heuristic labeling researches on it. Liu [3] proposed one method that first selected 15% positive examples and added them into U as 'spy' examples (identified by S) and then build a Naïve Bayes classifier using P-S and $U \cup S$ to extract the reliable negative examples. The shortage of this method is that it needed large amount of positive examples. Because Naïve Bayes is not a strong classifier for the texts, Li and Liu [6] further proposed a method based on Rocchio. It first built a Rocchio classifier to extract the reliable negative examples, but the accuracy was not high. In order to purify RN, it used clustering to partition RN into many clusters and built the classifier using each cluster and P. And then, it used these classifiers to identify the likely positive examples and removed them from RN. Yu [4] presented a framework PEBL. It first used 1-DNF algorithm to extract RN from U. The thought of 1-DNF algorithm is easy to understand, and it can extract the negative examples with the lowest error rate compared with the former two methods. The deficiency of this method is that it could not get enough reliable negative examples because of its strict constraint condition, so it is sensitive to the quality of P and it needs more effort in the second step.

DOI: 10.1002/cpe

The second step is to build the classifiers. Liu presented a method that ran the EM algorithm using the sets *P*, *RN* and *U* to build the final classifier until the algorithm converged [3]. SVM algorithm is used in PEBL [4] and Rocchio [6] to build the classifiers. Because there are few reliable negative examples can be extracted in the first step, the SVM needed to run iteratively. There is another fact that SVM is sensitive to the quality of the examples, so the last classifier may not be the best one. Different from two-step approach, Liu proposed a direct approach called biased-SVM [8]. Biased-SVM allowed the noise existing in *P*; it was following soft margin version of the biased SVM formulation that used two parameters to weight the positive errors and the negative errors, respectively.

The methods we introduced in the preceding texts have a major influence on the study of PU learning, and the performance of them can still be improved. On the basis of them, Fung [17] proposed a heuristic method called the Positive examples and Negative examples Labeling Heuristic aiming at extract not only negative examples (identified by *N*) but also positive examples from *U*, and it further enlarged the set *P* and the set *N* in order to increase the precision of PU learning. Similar to this work, Lu [18] proposed a refined method to do the PU learning by combining Rocchio and K-means algorithm, and this method also enlarged the set *P*. These two methods tried to extract the likely positive examples from the unlabeled examples and enlarged the given positive examples. Their works were novel and bring inspiration for the next work. Ke [19] came up with a method that injected a small amount of supervision information from users into the unsupervised framework probabilistic latent semantic analysis to solve the PU learning problem. The previously mentioned three methods could work well when the size of the set *P* is small.

In recent years, there are several methods with good performance that have been proposed. Claesen [16] presented an approach that regarded the PU learning problem as a supervised task with labeled noise in the negative set and used an ensemble of SVM models trained on bootstrap resamples of the training data for increased robustness against labeled noise. Liu [20] proposed a clustering-based method for collecting reliable negative examples. The idea of this method was to remove the probable positive examples from the set *U*, and the examples remained were regarded as reliable negative examples.

3. TRANSFER-1DNF METHOD FOR COLLECTING RELIABLE NEGATIVE EXAMPLES

Our method contains two parts: Transfer-1DNF and weighted voting iterative SVM (WV-ISVM). We improve the performance of the PU learning by improving the two parts, respectively. In this section, we describe the first part in details. Transfer_1DNF has two substeps: positive feature selection and reliable negative example selection. As mentioned in the preceding texts, positive feature selection is the key of Transfer_1DNF, and it is crucial to WV-ISVM. Our method is on the basis of 1-DNF that is used in PEBL framework. 1-DNF method first builds a positive feature set *PF* containing words that occur in the positive set *P* more frequently than in the unlabeled set *U* and then uses the *PF* to extract reliable negative examples. However, 1-DNF selects too many positive features including the ones which cannot stand for the positive category well because the method only considers the frequency of the feature. Therefore, 1-DNF cannot extract abundant reliable negative examples. For this reason, the first goal of our research is to select more accurate positive features.

3.1. Positive feature selection

In order to get more accurate positive features, we take the information which the features carry into account. Different from past methods, we do not need the feature distribution of the unlabeled set *U*. We aim to take a full consideration on the positive category. Through our analysis, positive category is not a single data island. Text data can be organized as a hierarchical structure. There are some relevant but not same categories with the positive category; we call them auxiliary categories. For example, if the positive category is 'basketball', then the auxiliary categories can be 'baseball', 'football' and so on. Moreover, all of them are the subcategories of the category 'sport'.

We analyze the similarities and the differences between the positive category and the auxiliary categories from the aspect of the feature. Take 20 Newsgroups [21] as examples, the percentage of

DOI: 10.1002/cpe

the features that the two different categories shared are reported in Table I. We find that the categories under the same top category share more features.

Inspired by the main idea of transfer learning, in this paper, we use some auxiliary categories to help us to find which features have the best presentation for the positive category. However, how can we get the auxiliary categories? We can regard the outdated data that have a relevance to the positive category as auxiliary categories. These data were used in other learning process, and they are often discarded because they do not follow the same distribution with the positive category. Although the outdated data cannot be used in the learning process directly, there is still some knowledge in the outdated data. For example, the corresponding feature distributions and the outdated models can be reused in the learning process. They can help us to select the features that are more relevant to the positive category.

Given two categories D1 and D2, the corresponding feature sets are $F_{d1}=\{f_{d11}, f_{d12}, \cdots, f_{d1n}\} \in R^n$ and $F_{d2}=\{f_{d21}, f_{d22}, \cdots, f_{d2m}\} \in R^m$, respectively. The two feature sets are mapped to a same feature space $F_d=\{f_{d1}, f_{d2}, \cdots, f_{dk}\} \in R^k$ $(k>m, n)$; this process is called 'dimension raising'. After this process, we need to reduce the dimensions of F_d because there has to be many useless features. We evaluate the features using information gain measure, and it is given by formula (1).

$$\begin{aligned} IG(f) &= H(C) - H(C|f) \\ &= -\sum_{i=1}^{n} P(C_i)\log_2 P(C_i) + P(t)\sum_{i=1}^{n} P(C_i|t)\log_2 P(C_i|t) \\ &\quad + P(\bar{t})\sum_{i=1}^{n} P(C_i|\bar{t})\log_2 P(C_i|\bar{t}) \end{aligned} \tag{1}$$

where $H(\cdot)$ is the entropy and $P(\cdot)$ is the probability.

This measure can evaluate the importance of a feature to the global dataset (including positive category and auxiliary categories). After dimension raising, we rank the features in a descending order according to their *IG* and remove the unimportant features.

The features remained are regarded as the necessary features to the positive category and the auxiliary categories. In our method, we further divide the positive features into two sets: the strong feature set and the weak feature set. The strong features are the features that have a strong presentation ability for the positive category, and the weak features are the features that have a weak presentation ability for the positive category.

Next, we select the strong features and the weak features for the positive category. We first use chi-squared (the formula 2 is given in the preceding texts) to measure the strength of dependence between

Table I. The statistics of the features shared between two categories.

Positive category	Other category	Percentage of the same features
comp.os.ms-windows.misc	comp.sys.ibm.pc.hardware	**36.8%**
comp.os.ms-windows.misc	rec.sport.hockey	27.1%
comp.os.ms-windows.misc	sci.space	34.2%
rec.motorcycles	rec.autos	**49.7%**
rec.motorcycles	talk.politics.mideast	46.7%
rec.motorcycles	sci.crypt	46.4%
sci.electronics	sci.space	**52.4%**
sci.electronics	talk.religion.misc	33.2%
sci.electronics	comp.sys.mac.hardware	42.6%
talk.politics.guns	talk.politics.mideast	**53.8%**
talk.politics.guns	comp.graphics	34.0%
talk.politics.guns	rec.sport.baseball	32.6%

The bold entries correspond to the largest similarities between a certain positive category and other categories. For example, we compare the percentage of the same features of the category 'comp.os.ms-windows.misc' with 'comp.sys.ibm.pc.hardware', 'rec.sport.hockey' and 'sci.space'. Because the categories 'comp.os.ms-windows.misc' and 'comp.sys.ibm.pc.hardware' share more features, the corresponding percentage of the same features is bold.

DOI: 10.1002/cpe

the feature and a specific category. If the value is high, it states that there is a strong relevance between the feature and the specific category.

$$\chi^2 = \sum_{i=1}^{n} \frac{(x_i - E)^2}{E} \tag{2}$$

where x_i is the observed probability and E is the expected probability.

Next, let *Strength* (·) be a function to compute the strength of a feature to the positive category.

$$Strength(f) = \sum_{i=0}^{|auxiliary|} \left(\chi^2(f, positive) + \chi^2(f, auxiliary_i)\right) \times \left(\chi^2(f, positive) - \chi^2(f, auxiliary_i)\right) \tag{3}$$

where f is a feature and $\chi^2(f, category)$ is the value of the importance of the feature f to the category.

We select the strong and the weak features of the positive category according to the *Strength* of the feature. Obviously, the strong features are more relative to the positive category than the weak features. If the *Strength* of the feature is positive and the value is greater than a support threshold α, the feature is considered as a strong feature. If the *Strength* of the feature is negative and the value is less than a threshold γ, we will abandon this feature because it means that the feature has a stronger relevance with the auxiliary categories. If the strength is in a range [γ, α], the feature is considered as a weak feature to the positive category. The parameter α and γ are defined according to the experiments. The details of the positive feature selection are described in Algorithm 1.

ALGORITHM 1: Positive Feature Selection
Input: the feature set of the positive category and the auxiliary category/categories, threshold α, γ
1. Get the feature space F_d by dimension raising
2. Initialize the candidate feature set F_C, the strong feature set F_S, the weak feature set F_W
3. **For** each feature f in F_dCalculate the *IG* of the feature f using formula (1)
End For
4. Rank the features according to *IG*(*f*) and put the top k into the set F_C
5. **For** each feature f in F_CCalculate the *Strength* of the feature f using formula (2)
End For
6. Rank the features according to the *Strength*
7. **Loop** each feature f in F_C
If *Strength (f)* $> \alpha$
Add the feature f into the set *Fs*
Else If *Strength (f)* $< \gamma$
Remove feature f from set F_C
End If
End If
End Loop
8. $F_W \leftarrow Fc\text{-}Fs$;
Output: the strong feature set *Fs* and the weak feature set F_W

3.2. Reliable negative example selection

The second substep is to selecting the reliable negative examples. We select the reliable negative examples using the strong and weak features. The strong features can provide the key information about the positive category, and the weak features can provide the complementary information about the positive category. Sometimes we cannot get abundant reliable negative examples only according to the feature distribution. Therefore, we collect a candidate negative example (*CN* for short) at the same time, and we can use them to supplement the *RN* when necessary.

Firstly, we use strong features to filter the unlabeled set U. The examples that are filtered out are dropped out. Secondly, we use the weak features to filter the U, the examples that are filtered out are considered as the candidate negative examples, and the examples that are not filtered out are

DOI: 10.1002/cpe

considered as the reliable negative examples. In order to guarantee that the *RN* set is sufficient, we set a minimum size ε of *RN*. The threshold ε is chosen through the experiments. If the *RN* set is insufficient, we select some examples from *CN* that are judged as negative ones by the existed model (or models) to expand the *RN* set. If there is more than one model, we can classify the example by the models using voting method. The majority class label is assigned to the example. The details are described in Algorithm 2.

ALGORITHM 2: Reliable Negative Example Selection

Input: positive examples set P, unlabeled examples set U, strong feature set F_S, weak feature set F_W, the minimum size ε of the reliable negative examples, the outdated classifier set M

1. Initialize the reliable negative example set *RN*, the candidate negative examples *CN*
2. $RN \leftarrow U$
3. **For** each example d in *RN*
If $\exists v \in F_S$ and $freq(v, d) > 0$ **then** // $freq(v,P)$: number of times that v
$RN \leftarrow RN - \{d\}$ // appears in P
End If
End For
4. **For** each example d in *RN*
If $\exists v \in F_W$ and $freq(v, d) > 0$ **then**
$RN \leftarrow RN - \{d\}$, $CN \leftarrow CN \cup \{d\}$
End If
End For
5. **While** $|RN| \leq \varepsilon$
For each example d in *CN*
If $(M_1(d) + \ldots\ldots M_{|M|}(d)) < 0)$ **then**
$RN \leftarrow RN \cup \{d\}$
End If
End For
End While
6. **Output:** the set *RN* and the set *CN*

4. BUILDING THE CLASSIFIERS BY APPLYING SUPPORT VECTOR MACHINE ITERATIVELY

The second step of PU learning is to build the classifiers. We first use vector space model to represent the documents in the training and the testing set, and we need to weight the features in the vector. A feature often plays a different role in the set P and the set *RN*, respectively. In order to reflect the different importance of the feature in the set P and the set *RN*, we adopt an improved term frequency-inverse document frequency method [22], term frequency inverse positive–negative document frequency (TFIPNDF), that is,

$$TFIPNDF = \begin{cases} f_{ik} \times \dfrac{P_i}{S_P} \times \log\left(\dfrac{N}{n_i}\right), & document\ k \in positive\quad examples \\ f_{ik} \times \dfrac{RN_i}{S_{RN}} \times \log\left(\dfrac{N}{n_i}\right), & document\ k \in negative\quad examples. \end{cases} \tag{4}$$

where f_{ik} is the number of the feature i occurs in the document k, P_i is the number of the feature i occurs in the positive set P, RN_i is the number of the feature i occurs in the negative set *RN*, S_p is the size of P, S_{RN} is the size of *RN*, N is the size of the entire training set and n_i is the number of the feature i occurs in the entire training set.

DOI: 10.1002/cpe

In order to ignore the influence of the different lengths between the documents, we use the normalized form that is defined as

$$TFIPNDF = \begin{cases} \dfrac{f_{ik} \times \dfrac{P_i}{S_P} \times \log\left(\dfrac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{rk} \times \dfrac{P_r}{S_P} \times \log\left(\dfrac{N}{n_r}\right)\right]^2}}, & document\ k \in positive\quad examples \\ \dfrac{f_{ik} \times \dfrac{RN_i}{S_{RN}} \times \log\left(\dfrac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{rk} \times \dfrac{RN_r}{S_{RN}} \times \log\left(\dfrac{N}{n_r}\right)\right]^2}}, & document\ k \in negative\quad examples \end{cases} \tag{5}$$

Next, we apply the SVM algorithm iteratively with P, RN and Q (U-RN) to build the classifiers, and Q is called the likely positive set. We use weighted voting method to construct our 'final classifier', so our method is called WV-ISVM. The details are given in Algorithm 3.

The basic idea is as follows. We first construct a validation set V that consists of the positive examples, the reliable negative examples and the candidate negative examples, and each example is assigned an initial weight. In each iteration, a new subclassifier C is constructed using P and RN, and we use C to classify the examples in Q. The examples that are classified as the negative examples make up the set W, and then, the W is removed from Q and added to RN. Next, we use validation set V to measure the performance of the subclassifier from two aspects. We use the positive and reliable negative examples to measure the precision of the subclassifier and use the candidate negative examples to measure the ability of the classifier that the classifier classifies the latent negative examples. After each measure, we adjust the weight of the example in V from two aspects. (1) The positive and the reliable negative examples are given a higher weight if they are not classified correctly, because these examples have a higher reliability and they need to be focused on. (2) The candidate negative examples are given a higher weight if they are classified correctly or given a lower weight if they are not classified correctly. That is, because these examples are with a low reliability, we need to lower their influence when they are misclassified. After each iteration, we calculate the error rate for the subclassifier. If the subclassifier is with a bad performance, we will drop this subclassifier or else we further calculate the confidence of this subclassifier. The iteration stops when no example in Q is classified as a negative one. The final 'classifier' is a combination of all the subclassifiers based on weighted voting method; the weight for each subclassifier is its confidence.

ALGORITHM 3: Weighted Voting Iterative Support Vector Machine

Input: positive examples set P, reliable negative examples set RN, candidate negative examples set CN, unlabeled examples set U

1. Extract some examples from P, RN and CN randomly to construct the validation set V and assign a initial weight $1/|V|$ for every examples in V
2. The examples in P are labeled 1, and the examples in RN are labeled -1
3. **Loop**

Construct a subclassifier C_i using P and RN, classify the U using C_i and put the examples that are classified as negative examples into W

If ($W = \varnothing$)

exit-loop

Else

$U \leftarrow U\text{-}W$

$RN \leftarrow RN \cup W$

End If

Test the C_i on the validation set V, calculate the error rate e_i of C_i

DOI: 10.1002/cpe

$$e_{\mathrm{i}} = \sum_{j=1^{|V|}} d^{i}_{w_j} \times \left| C_i(d_j) - Y(d_i) \right|$$

If ($e_i > 1/2$)
Drop the classifier C_i
Else
Calculate the factor δ_i
$\delta_i = e_i / 1 - e_i$
Update the weight of the examples in V

$$d^{i+1}_{w_j} = \begin{cases} d^{i+1}_{w_j} \times \delta_i & \text{if } w_j \in P \text{ or } w_j \in RN, \text{and } C_i(d_j) \neq Y(d_j) \\ d^{i+1}_{w_j} \times e_i & \text{if } w_j \in CN, \text{and } C_i(d_j) \neq Y(d_j) \\ d^{i+1}_{w_i} \times \delta_i & \text{if } w_j \in CN, \text{and } C_i(d_j) = Y(d_j) \end{cases}$$

End IF
End Loop
4. Calculate the confidence β_j of each available subclassifier C_j, and the confidence β_j of the dropped subclassifier is assigned 0.

$$\beta_j = \begin{cases} 1 - \dfrac{e_j}{\sum_{j=1}^{i} e_j}, & \text{if sub} - \text{classifier } C_i \text{ is existed} \\ 0, & \text{if sub} - \text{classifier } C_i \text{ is not existed} \end{cases}$$

5. Classify the document by weighted voting method

$$C(X) = \begin{cases} 1, & \text{if } \sum_{j=1}^{i} C_j(X) \times \beta_j > 0 \\ 0, & else \end{cases}$$

Output: the combination classifier $C(X)$

5. EXPERIMENTS AND RESULTS

In this section, we systematically evaluate our approaches including Transfer-1DNF and WV-ISVM on two datasets. We first introduce the datasets we used from the scale and the organization structure. Next, we give the evaluation metrics in common use. Finally, we show the results of the experiments and have a further discussion on the results. All the experiments are implemented in Java, and the system of the computer is Window 7 with Intel 5 processor and 8 GB memory.

5.1. Datasets and experimental settings

Our experiments are conducted on two collections: 20 Newsgroups and Reuters-21578 that are the popular datasets for text categorization research. We first give a brief introduction about them.

20 Newsgroups	The collection is collected by Lang from 20 different newsgroups. It contains seven top categories. Under the top categories, there are 20 subcategories, and each subcategory contains about 1000 articles. The hierarchical structure of this dataset is shown as Figure 2.
Reuters-21578	The collection has 21578 articles that are collected from Reuters newswire. It is publicly available at http://www.daviddlewis.com/resources/testcollections/reuters21578/. There are 135 categories, and among them, only 10 categories 9980 documents are frequently used. Moreover, they belong to five top categories. The hierarchical structure is shown as in Figure 3.

The similarity between the two datasets is that they can be organized as a hierarchical structure. We first split the dataset to generate the training and the testing sets. We select 50% of the documents as

DOI: 10.1002/cpe

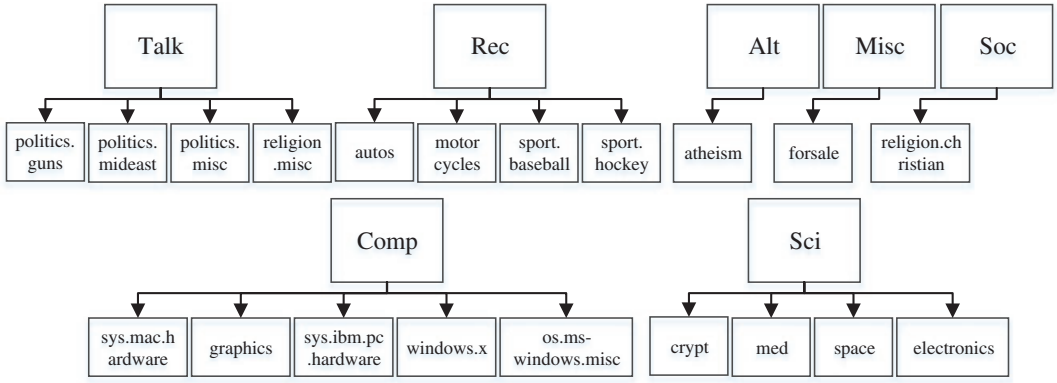

Figure 2. The structure of 20 Newsgroups dataset.

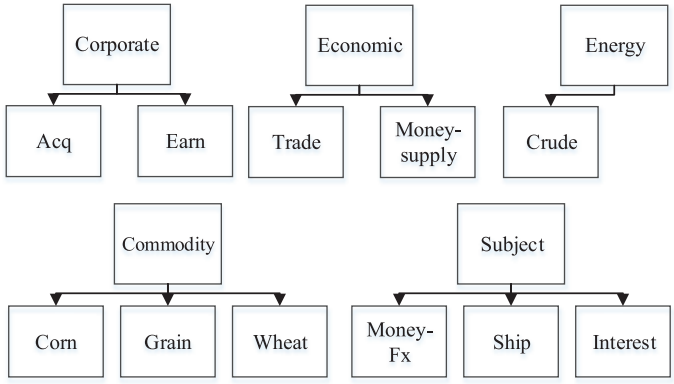

Figure 3. The structure of Reuter-21578 dataset.

training data and the remaining 50% as testing data. The datasets are not designed for our tasks originally, so we need to construct our training set in a way. Our training set contains three parts: the positive examples, auxiliary examples and the unlabeled examples. We select the positive examples and the auxiliary examples based on the hierarchical structure. The positive examples and the auxiliary examples should share the same top category. For example, if A is a top category that has four subcategories from A_1 to A_4, then we can choose one of them randomly as the positive category and the remaining subcategories are treated as auxiliary categories. The unlabeled example set consists of all the documents in training set excluding the ones under the category A. Tables II

Table II. The partition of the dataset Reuters-21578.

Positive class	Acq (mergers/acquisitions)	Corn	Interest (interest rates)	Money-supply (money supply)
Auxiliary class	Earn (earnings and earnings forecasts)	Grain wheat	Ship (shipping) money-Fx (money/foreign exchange)	Trade

Table III. The partition of the dataset 20 Newsgroups.

Positive class	comp.graphics	rec.autos	sci.crypt	talk.politics.guns
Auxiliary class	comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.sport.hockey rec.sport.baseball rec.motorcycles	sci.med sci.space sci.electronics	talk.politics.mideast talk.politics.misc talk.religion.misc

DOI: 10.1002/cpe

and III show the partition of the two datasets. Besides constructing training dataset and testing dataset, we randomly pick up 5% examples from training dataset to construct a validation set. It consists of positive examples, negative examples and candidate negative examples. Among them, positive, negative and candidate examples accounts for 40, 40 and 20% of the total validation set, respectively.

5.2. Evaluation metrics

For measuring the performance of the reliable negative example selection, we use the criterion *ERR* (the formula is as follows) defined in [22]. $RN(P_t)$ is the set of the positive examples in the reliable negative example set, and P_t is the set of positive examples in the unlabeled example set.

$$ERR = \frac{|RN(P_t)|}{P_t} \tag{6}$$

After building the classifier using training data, we will evaluate the classifier using a set of testing data. There are many evaluation metrics for text classification, for example, *Accuracy*, and it is defined as

$$Accuracy = \frac{C}{T} \tag{7}$$

where C is the number of the examples that are correctly classified, and T is the size of the testing dataset.

It is intuitive and easy to understand, but it is not suitable for our experiment. Because in each learning process we are only interested in one particular positive category, the positive category only accounts for about 10% of the whole training set. The classifier can easily achieve 90% accuracy by simply classifying every document as negative examples. So in this paper, we use *Precision* and *Recall* to measure the performance of the classifier. They are defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

where *TP* is the number of the correct classifications of the positive examples, *FP* is the number of the incorrect classification of the negative examples and *FN* is the number of incorrect classification of the positive examples [11].

Precision and *Recall* compare different classifiers from two aspects. *Precision* measures the accuracy that a classifier judges an example whether is a positive one, and *Recall* measures the ability that a classifier finds the positive examples. In general, high *Precision* is achieved by sacrificing the *Recall* and vice versa. Which of them is important depends on different applications. We use *F-measure* (the definition is as formula 10) to measure the performance of our method. *F*-measure is a single measure that balances the *Precision* and *Recall*.

$$F - measure = \frac{(\beta^2 + 1)Precison \times Recall}{\beta^2 \times Precison + Recall} \tag{10}$$

where β is a parameter to adjust the proportion between the *Precision* and *Recall*. For example, if β is greater than 1, *Recall* are considered more important than *Precision*. In our experiment, we assign β 1. When β is equal to 1, it is a harmonic mean of *Precision* and *Recall*, and it means that they have the same importance in our task.

DOI: 10.1002/cpe

5.3. Results

We first determine the parameters α, γ and ε through our experiments. There is an inverse relationship between the size of *PF* and the size of *RN*, and *ERR* is related to the size of *RN*, so we should make a trade-off between them. If *ERR* is high, it will affect the performance of the classifier because the SVM is sensitive to the noise data. Through the experiments, we find that we cannot select all the reliable negative examples only according to the feature distribution, so the goal of our method is to choose *RN* as many as we can with an error rate that we can bear. As is shown in Table IV, through the experiments, we assign the minimum size ε 1210, and the parameter α is assigned 0.00002. When the *Strength* of the feature is less than 0, the feature is definitely more relative to the auxiliary categories than to the positive category, so the γ is assigned 0.

In order to evaluate the effectiveness of our proposed method Transfer-1DNF, we conduct our experiments on the two datasets and compare the Transfer-1DNF with 1-DNF [4] and 1DNFII [23] from two aspects: the size of the *RN* and the *ERR* criterion. In the experiments, we randomly pick up 15% examples from the set *P* as spy examples and put them in the unlabeled examples set *U*, and we run the 1DNFII on $\lambda = 0.2$ setting. The number of the reliable negative examples that are extracted by the three methods and the corresponding *ERR* are shown in Tables V and VI. From the results, we find that Transfer-1DNF can extract more examples than the other two methods, and the *ERR* of our method is lower than 1-DNFII and higher than 1-DNF. Although the *ERR* of 1-DNF is approximately 0, it only extracts a small amount of reliable negative examples from *U*.

Next, we use the positive examples and the reliable negative examples we just extracted to build the classifiers. We first do some pre-processing work. Because the feature space of the documents contains tens of thousands of features, we need to reduce the dimension of the feature space. The stop words and

Table IV. The performance of selecting RN at different parameter α.

α	The size of *PF*	The size of *RN* (ε)	*ERR*
0.0002	2000	1839	0.0632
0.0001	2234	1664	0.0407
0.00005	2587	1497	0.0252
0.00002	**2776**	**1210**	**0.0147**
0.00001	2913	971	0.0091

The bold entries give the reason why we choose the thresholds of ε and α. We should balance the relationship between the size of RN and ERR. The bold entries are the best choice of the thresholds.

Table V. The number of reliable negative examples and error rate on Reuters-21578.

		Acq	Corn	Interest	Money-supply
1-DNF	RN	209	213	270	252
	ERR	0.0014	0	0	0
1-DNFII	RN	907	769	1120	1195
	ERR	0.025	0.0175	0	0
Transfer-1DNF	RN	1287	1326	1392	1759
	ERR	0.0097	0.0115	0.0158	0.0136

Table VI. The number of reliable negative examples and error rate on 20 Newsgroups.

		comp.graphics	rec.autos	sci.crypt	talk.politics.guns
1-DNF	RN	114	109	98	126
	ERR	0	0	0	0
1-DNFII	RN	429	382	403	516
	ERR	0.0176	0.0132	0.0141	0.019
Transfer-1DNF	RN	1615	1210	1523	1813
	ERR	0.0157	0.0124	0.0137	0.0171

DOI: 10.1002/cpe

Table VII. The Recall and Precision of the four methods on Reuters-21578.

		Acq	Corn	Interest	Money
WV-ISVM	Recall	0.9781	0.9338	0.9388	0.9389
	Precision	0.9650	0.9262	0.9237	0.9260
WVC (1-DNFII)	Recall	0.9445	0.8925	0.9102	0.9139
	Precision	0.9505	0.8852	0.8991	0.8993
PEBL	Recall	0.9263	0.8760	0.8897	0.8833
	Precision	0.9222	0.8617	0.8790	0.8712
TSVM	Recall	0.9145	0.8925	0.8897	0.9139
	Precision	0.9063	0.8780	0.8861	0.8939

Table VIII. The Recall and Precision of the four methods on 20 Newsgroups.

		comp.graphics	rec.autos	sci.crypt	talk.politics.guns
WV-ISVM	Recall	0.8955	0.8838	0.8891	0.8718
	Precision	0.8790	0.8663	0.8644	0.8500
WVC (1-DNFII)	Recall	0.8647	0.8343	0.8689	0.8516
	Precision	0.8516	0.8120	0.8489	0.8424
PEBL	Recall	0.8476	0.8333	0.8554	0.8189
	Precision	0.8264	0.8115	0.8386	0.7996
TSVM	Recall	0.8339	0.8434	0.8605	0.8388
	Precision	0.8144	0.8253	0.8479	0.8164

the features with lower *term frequency* (that is less than 5) are removed. For comparison, we used toolkit LIBSVM [24] to implement the four methods: WV-ISVM, weighted voting classifier (WVC) [23], PEBL and traditional SVM (TSVM) on the two datasets, respectively. The parameters of SVM we use are standard.

Table VII and Table VIII show the *Recall* and *Precision* of the four methods on the two datasets, respectively. From the results, we can find that our method outperforms the other three methods. In order to look at the results more intuitively, we further report the results with F-measure that is a standard measure for binary classification. Figure 4 plots the F-measure of WV-ISVM, WVC, PEBL and TSVM over two datasets. The results indicate that the performance of WV-ISVM is good. The F-measure of our method WV-ISVM is higher than the other three methods on both two datasets.

Finally, we give a detailed analysis about the results. Compared with PEBL, and WVC, we have an improvement for each step. From the results shown in the preceding texts, we find that the first step is the key to the second step. Both 1-DNF and 1-DNFII select the positive features by considering the frequency of the features. 1-DNF only considers the frequency of the feature that occurs in P and U, so that the positive feature contains many features that cannot stand for the positive category. These inaccurate positive features can filter out many reliable negative examples. For example, if the frequency of a feature f is 0.0011 in P and 0.0010 in U, 1-DNF will add the feature f into the positive feature set. Obviously, we cannot regard the feature f as a positive feature. 1-DNFII improved the 1-DNF by adding a limiting condition that the frequency of the feature in P should be greater than a fixed threshold. Although 1-DNFII can reduce the dimension of the positive features to a certain extent, it cannot extract abundant reliable examples because it still only considers the frequency. Different from them, Transfer-1DNF take the information that the features carry into account, so it can select more accurate positive features related to positive category. Transfer-1DNF only uses half of the positive features that 1-DNF uses, and it can efficiently extract large amount of the reliable negative examples from unlabeled dataset with a low error rate.

In the second step, all the methods except the TSVM apply SVM iteratively. From the results, we find that TSVM can get relatively better performance with no iteration, but TSVM needs labeled negative examples. It causes the label and is time-consuming. PEBL uses the last classifier as its final classifier, but the last one may not be the best one. That is, because PEBL extracts too few

DOI: 10.1002/cpe

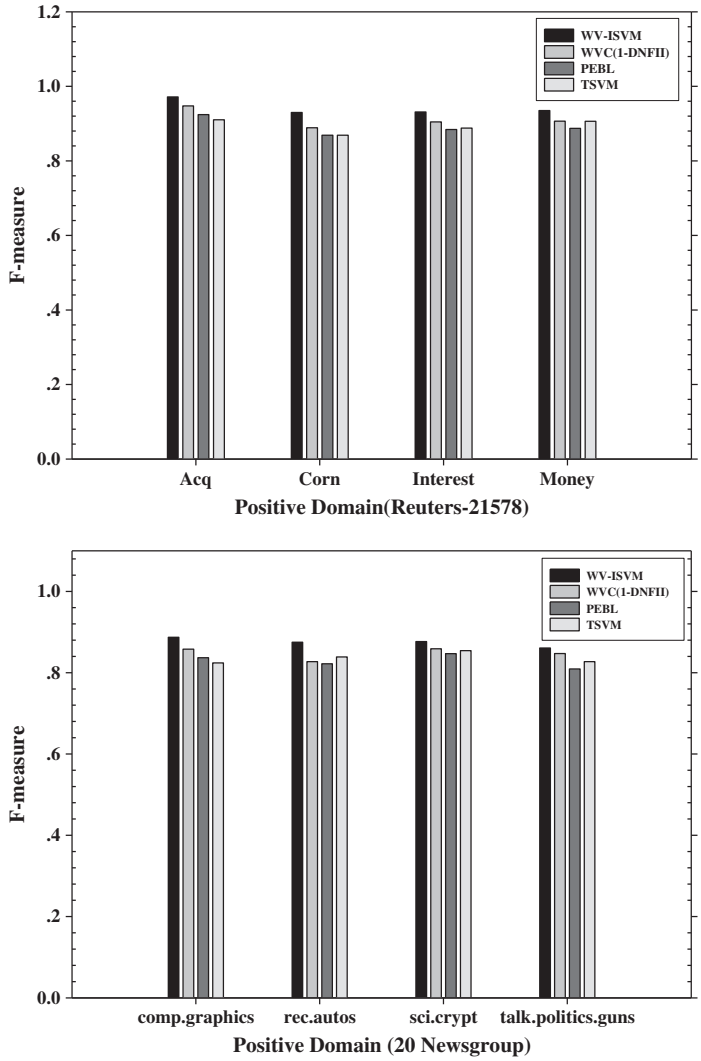

Figure 4. The *F*-measure of the three classifying methods for each topic on two datasets.

reliable negative examples in the first step, the performance of the last classifier is unstable. WVC constructs a group of subclassifiers, and it finally classifies the examples using weighted voting method; the weight is the precision of each subclassifier. Our proposed method WV-ISVM also uses weighted voting method; the weight in our method is the confidence of each subclassifier. We calculate the confidence of each classifier on a validation set. Because the validation set concludes the positive examples, the reliable negative examples and the candidate examples, it can assess the subclassifier in a comprehensive way.

6. CONCLUSION

In this paper, we first present a novel heuristic learning method for extracting reliable negative examples inspired by the idea of transfer learning. We aim to take full advantage of the outdated data. Different from the previous methods, we select positive features not simply according to the frequency of the features. We take the information which the features carry into account. We evaluate the strength of the features and choose the strong and the weak features according to the feature strength. Then, we extract the reliable negative examples using the strong and the weak features. Besides the reliable negative examples, we also extract some candidate negative examples.

DOI: 10.1002/cpe

The candidate negative examples have two functions in our learning process. We use them to supplement the *RN* when *RN* is less than the minimum size, and we also use them to evaluate the subclassifier in each iteration. Finally, we built a WVC by applying the SVM iteratively. We compare our method with two baseline algorithms PEBL and WVC. The results show that our method can extract more reliable negative examples with lower error rates and our classifier outperforms the other two state-of-the-art classifiers.

With the advent of the Big Data, the data is getting more complex and larger. In this situation, the premise of the traditional machine learning methods cannot be satisfied in usual. The Big Data era has brought a great challenge to the traditional machine learning methods. There will also be more and more 'outdated' data and outdated models. However, the outdated data does not mean that they are valueless. To the contrary, there are much knowledge not mined in the outdated data. So, how to reuse them has become an interesting and meaningful task. Transfer learning is a kind of method to solve this problem. In the future work, we will further study how to use the outdated data to address some important issues such as text classification, image classification and sensor network-based localization.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China under grant nos. 60903098 and 60973040 and Graduate Innovation Fund of Jilin University under no. 201540.

REFERENCES

1. Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys* 2002; **34**(1):1–47.
2. Han JW, Kamber M, Pei J. *Data Mining: Concepts and Techniques* (3rd edn). Morgan Kaufmann: Burlington, 2011; 360–361.
3. Liu B, Lee WS, Yu PS, *et al.* Partially supervised classification of text documents. In *Proceeding ICML '02 Proceedings of the Nineteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2002; 387–394.
4. Yu H, Han J, Chang K C C. PEBL: positive example based learning for web page classification using SVM. *Proceeding KDD '02 Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, ACM New York, NY, USA 2002; 239-248.
5. Manevitz LM, Yousef M. One-class SVMs for document classification. *Journal of Machine Learning Research* 2002; **2**(2):139–154.
6. Li X, Liu B. Learning to classify texts using positive and unlabeled data. In *Proceeding IJCAI'03 Proceedings of the 18th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2003; 587–592.
7. Lee W S, Liu B. Learning with positive and unlabeled examples using weighted logistic regression. *Machine Learning*, *Proceedings of the Twentieth International Conference*. 2003, Washington, DC, USA, 2003; 448-455.
8. Liu B, Dai Y, Li X, *et al.* Building text classifiers using positive and unlabeled examples. In *Proceeding ICDM '03 Proceedings of the Third IEEE International Conference on Data Mining*. IEEE Computer Society: Washington, DC, USA, 2003; 179–186.
9. Li H, Bing L, Mukherjee A, Shao J. Spotting fake reviews using positive-unlabeled learning. *Computación y Sistemas* 2014; **18**(3):467–475.
10. Fusilier DH, Montes-y-Gómez M, Rosso P, *et al.* Detecting positive and negative deceptive opinions using PU-learning. *Information Processing and Management* 2014; **51**(4):433–443.
11. Liu B. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, Vol. **81-82** (2nd edn). Springer: Heidelberg Dordrecht London: New York, 2011; 194–195.
12. Pan SJ, Yang Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 2010; **22** (10):1345–1359.
13. Vert R, Vert JP. Consistency and convergence rates of one-class SVMs and related algorithms. *Journal of Machine Learning Research* 2006; **7**(5):817–854.
14. Larry M. Manevitz, MY. Document classification on neural networks using only positive examples. *SIGIR '00 Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000; 304-306.
15. Schölkopf B, Platt JC, Shawe-Taylor J, *et al.* Estimating the support of a high-dimensional distribution. *Neural Computation* 2001; **13**(7):1443–1471.
16. Claesen M, Smet FD, Suykens JAK, Moor BD. A robust ensemble approach to learn from positive and unlabeled data using SVM base models. *Neurocomputing* 2015; **160**:73–84.
17. Fung GPC, Yu JX, Lu H, *et al.* Text classification without negative examples revisit. *Knowledge and Data Engineering, IEEE Transactions on* 2006; **18**(1):6–20.

DOI: 10.1002/cpe

18. Lu F, Bai Q. Semi-supervised text categorization with only a few positive and unlabeled documents. *Biomedical Engineering and Informatics* (*BMEI*), *2010 3rd International Conference on*, 2010, IEEE, Yantai, 2010; 7: 3075-3079.
19. Zhou K, Gui-Rong X, Yang Q, *et al.* Learning with positive and unlabeled examples using topic-sensitive PLSA. *Knowledge and Data Engineering, IEEE Transactions on* 2010; **22**(1):46–58.
20. Liu L, Peng T. Clustering-based method for positive and unlabeled text categorization enhanced by improved TFIDF. *Journal of Information Science and Engineering* 2014; **30**(5):1463–1481.
21. 20 Newsgroups. (Available from: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.) [Accessed on 9 May 2015]
22. Peng T, Liu L, Zuo W. PU text classification enhanced by term frequency-inverse document frequency-improved weighting. *Concurrency and Computation: Practice and Experience* 2014; **26**(3):728–741.
23. Peng T, Zuo W, He F. SVM based adaptive learning method for text classification from positive and unlabeled documents. *Knowledge and Information Systems* 2008; **16**(3):281–301.
24. LIBSVM -- A Library for Support Vector Machines. (Available from: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.) [Accessed on 14 September 2015]

DOI: 10.1002/cpe