Text Classification Using Clustering

Antonia Kyriakopoulou and Theodore Kalamboukis

Department of Informatics, Athens University of Economics and Business
76 Patission St., Athens, GR 104.34
{tonia, tzk}@aueb.gr

Abstract. This paper addresses the problem of learning to classify texts by exploiting information derived from both training and testing sets. To accomplish this, clustering is used as a complementary step to text classification, and is applied not only to the training set but also to the testing set. This approach allows us to estimate the location of the testing examples and the structure of the whole dataset, which is not possible for an inductive learner. The incorporation of the knowledge resulting from clustering to the simple BOW representation of the texts is expected to boost the performance of a classifier. Experiments conducted on tasks and datasets provided in the framework of the ECDL/PKDD 2006 Challenge Discovery on personalized spam filtering, demonstrate the effectiveness of the proposed approach. The experiments show substantial improvements on classification performance especially for small training sets.

1 Introduction

Text classification is one of the first applications of machine learning, that applies to the general problem of supervised inductive learning: given a set of training documents, classified to one or more predefined categories, learn to automatically classify new documents. Automated text classification has been used in a number of different applications: automatic indexing, content management, filtering, and routing, word sense disambiguation, and Yahoo!-style search space categorization [1, 20]. A plethora of techniques have been developed for text classification, including Nearest-Neighbours [29], Regression [30], Neural Networks [28, 14], Naive Bayes [12], Decision Trees [27], and Support Vector Machines [26, 8]. In most cases, the classification algorithms require sufficient training data in order to generalize well on unseen documents. However, the generalization using labeled examples is an extremely costly and time-consuming activity. The need for classifiers that can learn from small training samples is imperative. This is an area of active research and several experiments have been conducted to boost conventional classifiers' performance, by combining supervised learning with semi-supervised or unsupervised, using techniques such as co-training [5, 13], active learning [25, 19], and transductive SVMs [10].

In traditional supervised classification an inductive learner is first trained on a training set, and then is called to classify a testing set, about which it has no prior knowledge. An ideal situation would be for the classifier to have information about the distribution of the testing examples before it classifies them. This remark motivates the work included in this paper. In this vein, the goal of this paper is to deal with the problem of learning from training sets of different sizes, by exploiting information derived from clustering the whole dataset (both training and testing examples), and embodied in it in the form of *meta-information.*

Clustering has been used in the literature of text classification either as an alternative approach to term selection for dimensionality reduction or as a technique to enhance the training set. In the second case, clustering is used to discover a kind of "structure" in the training examples and expand the feature vectors with new attributes extracted from clusters. Also, it is used to augment a small number of labeled examples with unlabeled examples by propagating label information to the unlabeled data according to clustering results on both labeled and unlabeled data. Several approaches of clustering have been proposed in these areas.

In [2], *class-distributional clustering* [15] is applied as a feature selection method in a text classification context using a Naive Bayes classifier. Words are clustered into groups based on the distribution of class labels associated with each word. In [21], the *information bottleneck (IB)* method [24] is used to find word-clusters that preserve the information about the categories as much as possible. These clusters are used to represent the documents in a new, low dimensional feature space and a Naive Bayes classifier is applied. Accordingly, [3, 4] also use the IB framework to generate a document representation in a word cluster space instead of word space, where words are viewed as distributions over document categories. [6, 7] propose an information-theoretic divisive algorithm for word clustering and apply it to text classification. Classification is done using word clusters instead of simple words for document representation. [22, 23] adopt a more complicated approach, applying *two-dimensional clustering* to text classification. They cluster the training examples in order to deal with problems where the texts in a category are not generated from an identical probability distribution, and also they cluster the words/features of these examples in order to avoid the data sparseness problem. The evaluation is done using Naive Bayes and SVM classifiers on Reuters-21578 corpus and shows a superiority of their algorithm over class-distributional clustering and word clustering.

The idea of using clustering as a technique to enhance the training set is pursued in many works too. In [16], an algorithm is described that uses unlabeled data, independent from the testing set, to improve text classification performance. The algorithm applies clustering to labeled and unlabeled data, and introduces new features extracted from those clusters to the patterns in the labeled and unlabeled data. They evaluate the method using SVM classifiers on Reuters-21578, 20Newsgroups, and WebKB corpora, and find significant improvements in their classification performance. In [17], the technique presented above is combined with co-training. The algorithm trains two predictors in parallel, with one predictor labeling the unlabeled data for training the other in the next round. The predictors are SVMs, one trained using data from the original feature space, and the other trained with new features that are derived from clustering both labeled and unlabeled data. This new input feature space creates an alternative view of the data, which is used for co-training, using the same supervised learning algorithm that is used for the original feature space. The evaluation of the method using SVM classifiers on Reuters-21578, 20Newsgroups, and WebKB corpora confirm previous findings. The *clustering based text classification (CBC)* approach [31], adopts a different way of exploiting the unlabeled data. According to this approach, labeled training data and unlabeled data are first clustered. Some of the unlabeled data are then labeled based on the clusters obtained, i.e. the labels of the labeled data are propagated to the unlabeled data that are closest to the cluster centroids. Discriminative classifiers are subsequently trained with the expanded labeled dataset. Their experimental results on 20Newsgroups, Reuters-21578 and Open Directory Project (ODP), demonstrated that CBC outperforms existing algorithms, such as TSVMs and co-training, especially when the size of the labeled dataset is very small.

The works of [16, 31] could be considered most relevant ones to our approach. However, they use clustering in order to create a better training set, without looking into the testing set as we do.

In this article, an algorithm that combines supervised and unsupervised classification is proposed. In the unsupervised case, the aim is to extract a kind of "structure" from a given sample of objects, or to rephrase it better to learn a concise representation of these data. The reasoning behind this is that if some structure exists in the objects, it is possible to take advantage of this information and find a short description of the data. In our approach, given a classification problem, the training and testing examples are both clustered before the classification step, in order to extract the "structure" of the whole dataset, exploiting the dependence or association between index terms and documents. The structure extracted from the dataset is "translated" in such a way that each cluster is represented by one representative. This concise representation of the whole dataset is incorporated in the existing data representation; each object is assigned the corresponding cluster id using appropriate artificial *meta*-features. It is expected that the use of prior knowledge about the nature of the testing set will help in building a more efficient classifier for this set.

The paper is organized as follows. Sections 2 and 3 describe the proposed algorithm. In section 4, the experimental settings, i.e. the datasets, the evaluation metrics and the experimental results are presented. Finally, section 5 concludes with a summary of the work and future research.

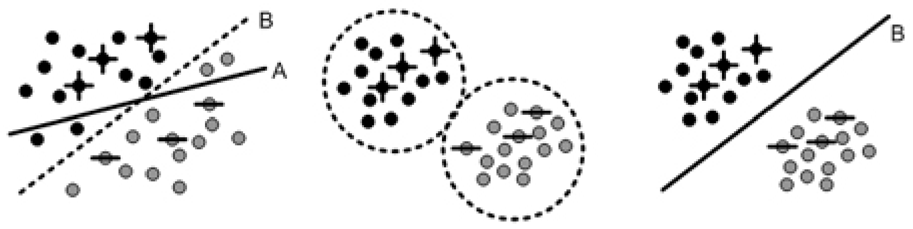
2 Classification with Clustering

In this section, we give the intuition of the proposed algorithm in order to understand how clustering prepares the ground for classification. The algorithm consists of the following steps:

1. Clustering step: to cluster both the training and testing set.
2. Expansion step: to augment the dataset with *meta*-features originated from the clustering step.
3. Classification step: to train a classifier with the expanded dataset.

Figure 1 gives an insight into our approach. The labeled examples of the training set are denoted with + and − signs, while the unlabeled examples of the testing set are denoted with dots.

A classifier trained with the given training examples will probably find hyperplane A instead of the desirable hyperplane B, as shown in Fig. 1(a). In Fig. 1(b), both datasets (training and testing) are clustered into two non-overlapping clusters. In the ideal case, the two clusters contain the positive and negative examples of the whole data set respectively. Then, corresponding *meta*-features are propagated to the existing feature vectors, and all feature vectors inside the same cluster are augmented with the same *meta*-feature. The dataset is transformed into a new coordinate system. Since feature vectors inside the same cluster are augmented with the same attribute-value pair, these vectors are now closer to one another resulting to an increase in the dataset's density, illustrated in Fig. 1(b). Increasing the inter-cluster distance consequently leads to the maximal margin hyperplane B, as shown in Fig. 1(c). In a way, the classifier is tuned to the testing set and the classification efficiency is expected to improve. Intuitively, the classifier with the largest margin will give lower expected risk, i.e. better generalization.



(a) Classification with the original data (b) Expanded dataset after clustering (c) Classification with the new expanded dataset

Fig. 1. An insight into the algorithm proposed

3 The Algorithm

In this section, we present our algorithm in more details. Following the traditional IR approach, we consider a k-class categorization problem, ($k = 1$ in the case of the spam filtering problem), with a labeled l-sample $\{(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)\}$ of feature vectors $\vec{x_i} \in \mathcal{R}^n$, and corresponding labels $y_i \in \{1, \ldots, l\}$, and an unlabeled m-sample $\left\{\vec{x_1^*}, \ldots, \vec{x_m^*}\right\}$ of feature vectors, where $m \gg l$. In their original representation – that given in the framework of the Challenge – the datasets consist of feature vectors (documents) whose terms (features) are valued with their *term frequency*, $TF(w_i, \vec{x})$, i.e. the number of times term w_i occurs in the document $\vec{x}$. However, in our implementation the *TFIDF* model [18] is used, defined by

$$W(w_i) = TF(w_i, \vec{x}) * IDF(w_i) \tag{1}$$

where

$$IDF(w_i) = \log_2 \left(\frac{|X|}{DF(w_i)} \right) \tag{2}$$

is the *inverse document frequency* $IDF(w_i)$ with $|X|$ total number of documents in the training set, and *document frequency*, $DF(w_i)$, the number of documents that contain the term w_i. All feature vectors are normalized to unit length. For the classification step, the terms that appear only in the testing set but not in the training set are discarded.

The CLUTO^{TM} Clustering Toolkit [11] is used, and a divisive clustering algorithm with repeated bisections is applied. In this method, the disired k-way clustering solution is computed by performing a sequence of $k-1$ repeated bisections. The dataset is first clustered into two groups, then one of these groups is selected and disected further. This process continuous until the desired k number of clusters is found. During each step, the cluster is bisected so that the resulting 2-way clustering solution optimizes the internal criterion function

$$\max \sum_{i=1}^{k} \sqrt{\sum_{\vec{x}_v, \vec{x}_u \in S_i} sim(\vec{x}_v, \vec{x}_u)} \tag{3}$$

where S_i is the set of documents assigned to the i^{th} cluster, and $sim(\vec{x}_v, \vec{x}_u)$ is the similarity between documents $\vec{x}_v$ and $\vec{x}_u$. The generated clusters are non-overlapping.

After the clustering step, each cluster contributes one *meta*-feature to the feature space of the training and testing sets: given the total n features that are used in the representation of the $l+m$ feature vectors, and the k clusters derived from the clustering step, create *meta*-features $x_{n+1}, \ldots, x_{n+k}$. A document $\vec{x}$ that belongs to cluster C_j is characterized by the *meta*-feature x_{n+j}. The weight of that *meta*-feature is computed applying the *TFIDF* model to the clusters. Considering that each document in the cluster contains this *meta*-feature, its *term frequency* $TF(x_{n+j}, \vec{x}) = 1$ and its *inverse document frequency* is defined accordingly

$$IDF(x_{n+j}) = \log_2 \left(\frac{|X|}{|C_j|} \right) \tag{4}$$

Finally, on the classification step the SVM^{light} implementation of SVMs and TSVMs is used [9, 10]. A binary classifier is constructed for each user's *expanded* dataset, a linear kernel is used and the weight C of the slack variables is set to default.

The proposed algorithm is summarized in Table 1.

Table 1. The proposed algorithm.

Clustering step	
Input:	training examples $(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)$
	testing examples $\vec{x_1^*}, \ldots, \vec{x_m^*}$
	$k =$ desired number of clusters
	Use a clustering algorithm to cluster all examples
Output:	k cluster ids
Expansion step	
Input:	training examples $(\vec{x_1}, y_1), \ldots, (\vec{x_l}, y_l)$
	testing examples $\vec{x_1^*}, \ldots, \vec{x_m^*}$
	k cluster ids
	Create additional *meta*-features for the vectors of *all* the examples.
	Each cluster corresponds to one new *meta*-feature.
	Given the total n features that produce the $l + m$ example vectors, and the k clusters derived from the clustering step, create *meta*-features $x_{n+1}, x_{n+2}, \ldots, x_{n+k}$. The values of these new features are defined by
	$W(x_{n+j}) = \left\{ \begin{array}{cc} \log_2\left(\frac{\lvert X\rvert}{\lvert C_j\rvert}\right) & \text{if } \vec{x} \in C_j \text{ for } j = 1, \ldots, k \\ 0 & \text{otherwise} \end{array} \right\}$
Output:	expanded training examples $(\vec{x_1'}, y_1), \ldots, (\vec{x_l'}, y_l)$
	expanded testing examples $\vec{x_1^*}', \ldots, \vec{x_m^*}'$
Classification step	
Input:	expanded training and testing examples of previous step.
	Train a SVM/TSVM classifier based on the *expanded* training examples.
	Classify the *expanded* testing examples.
Output:	predicted labels of the testing examples $y_1^*, \ldots, y_m^*$

A generalization of our algorithm includes the addition of more than one *meta*-features for each cluster. In this case, if f is the desired number of *meta*-features to be added per cluster, then an example $\vec{x}$ that belongs to cluster C_j is expanded with meta-features $x_{n+(j-1)*f+1}, \ldots, x_{n+(j-1)*f+f}$. Experiments that have been conducted show that expanding an example with more than one *meta*-features has a positive effect to classification. However, these experiments are beyond the scope of this paper.

4 A Performance Study

4.1 Experiment Settings

The empirical evaluation is done in two tasks, created and published in the framework of EMLC/PKDD 2006 Challenge Discovery[1]. The goal in both tasks is to make a personalized spam filter for a single user's inbox that correctly classifies its emails as *spam* or *non-spam*. In Task A, each classifier is made using the available training examples and each inbox separately, taking into account the user's inbox specific characteristics. In Task B, the learning algorithm is supposed to generalize over the different users in such a way that data from the other users may be utilized in order to enhance classification performance. In our experiment, however, both tasks are used in the same way, that defined for Task A. The reason for this is that we want to explore the effect of our technique when different sizes of training samples are used; Task A contains 4.000 training examples, whereas Task B only 100 training examples. The evaluation criterion prescribed by the competition is the AUC value. AUC values are computed for each user separately and average over all users.

4.2 Results and Further Discussion

To provide a baseline for comparison, results from the standard SVM and transductive SVM classifiers are also presented.

Preliminary results from experiments conducted on three widely used corpora (Reuters, Ohsumed, and WebKB) have shown an increase of performance of classification when the number of clusters is equal to the number of the predefined classes. In traditional classification tasks it can be assumed that the classes correspond to topics, and there is a one-to-one correspondence between the topic and the class under which the data are classified. Moreover, the examples of a class are clustered together which is logical since they share the same word distribution. So we can assume that there is a one-to-one correspondence between classes, topics and clusters, and use this information to define the desired number of clusters. In spam filtering we can't make such safe assumptions. *Spam* emails can deal with many different topics, there is a one-to-many correspondence between the class *spam* and the topics of the examples that fall under it. The obvious number of clusters to select is two: one cluster with the *spam* emails and one cluster with the *non-spam*. But we loosen this assumption based on the rationale that not all people consider an email as *spam* or *non-spam*. It is suggested that *spam* emails should be similar in both the public domain emails of the training set and the users' inboxes (testing set), while the *non-spam* should differentiate. According to this assumption, the number of clusters is chosen to be equal to *three*: one cluster for the common *spam* emails, one cluster for the common *non-spam* emails and one cluster for the rest.

[1] The task specific number of emails and inboxes, and additional information about the settings of the Challenge Discovery, can be found in http://www.ecmlpkdd2006.org/challenge.html

Table 2 gives the results for Task A. The proposed method leads to an improvement in performance on all users, raising the average AUC by 6.6% when the SVM classifier is used with clustering and by 3.2% when the TSVM classifier is used accordingly.

Table 2. Average AUC for the users of Task A.

Users	**Standard SVM**	**Clustering +SVM**	**Standard TSVM**	**Clustering +TSVM**
user00	84.72	93.26	89.44	95.25
user01	89.10	96.65	94.43	97.28
user02	94.70	96.58	98.92	99.40
Average AUC	**89.51**	**95.50**	**94.26**	**97.31**

The results on the Task B datasets in Table 3 show a 3.2% improvement on AUC over the standard TSVM classifier. Due to the small number of the training and testing examples for this task (100 and 400 examples respectively), no SVMs were used in the classification step. In the last column of Table 3, we present the best results obtained in terms of AUC after several runs of the algorithm with various numbers of clusters and *meta*-features used. These results are for demonstration purposes only and they were possible after the release of the data with their true labels by the Challenge organizers. The numbers of clusters, k, and *meta*-features, f, used in these experiments are mentioned in the parentheses. The results reveal that there is still room for improvements in performance, which is currently under investigation.

One limitation of our algorithm is that with the constant arrival of new emails, the same procedure of clustering, *meta*-feature addition, and classification, should be applied again for the whole dataset, a rather time consuming, and computationally expensive process. A suggestion would be to use incremental clustering instead of the static clustering algorithm used now. Incremental clustering is a method that deals with the problem of updating clusters without frequently performing complete reclustering. This would be a more suitable way for maintaining clusters in the typical, dynamic environment of spam filtering.

Another issue about our algorithm is its rather naive approach to clustering that may not capture all the *meta*-information possible hidden in the dataset. More sophisticated clustering methods have been proposed in the literature that focus on incorporating prior knowledge into the clustering process; conceptual clustering, topic-driven clustering [32], just to name a few. These methods are based in the idea that it is possible to use explicitly available domain knowledge to constrain or guide the clustering process. In our case, the class labels of the training set can constitute the domain knowledge and be used as guidance to a clustering algorithm.

Another issue that needs to be discussed is the representation of the extra knowledge derived from clustering, i.e. the representation of the clusters. The

Table 3. Average AUC for the users of Task B.

Users	Standard TSVM	Clustering +TSVM	Clustering +TSVM (best)
user00	96.76	97.36	98.19 ($k = 2, f = 12$)
user01	91.75	96.67	98.50 ($k = 5, f = 5$)
user02	97.34	97.09	99.93 ($k = 5, f = 5$)
user03	97.11	99.09	99.73 ($k = 5, f = 3$)
user04	94.91	96.44	97.99 ($k = 10, f = 5$)
user05	82.09	95.74	97.31 ($k = 3, f = 3$)
user06	82.91	90.87	93.06 ($k = 4, f = 5$)
user07	98.19	97.65	99.24 ($k = 3, f = 15$)
user08	98.95	99.16	99.77 ($k = 5, f = 1$)
user09	98.18	96.85	99.25 ($k = 2, f = 5$)
user10	92.56	92.99	96.16 ($k = 10, f = 15$)
user11	92.19	94.73	96.73 ($k = 4, f = 5$)
user12	87.77	88.92	92.38 ($k = 4, f = 15$)
user13	92.32	90.14	97.53 ($k = 4, f = 3$)
user14	78.98	92.44	99.28 ($k = 3, f = 15$)
Average AUC	**92.13**	**95.08**	**97.67**

representation schemes where a cluster of points is represented by their centroid or by a set of distant points in the cluster are the most popular ones. It appears that a different cluster representation scheme and its readjustment to our model's terrain might reflect the structure of more complex datasets, in a more efficient way. We hope that this can be further evaluated in future works.

5 Conclusions

This paper has introduced a new way to combine clustering with classification. We presented experimental results on datasets given in the framework of the ECML/PKDD Discovery Challenge 2006 on spam filtering. On all the collections the clustering approach combined with a SVM/TSVM classifier outperformed the standard SVM/TSVM classifier.

Possible extensions and improvements of our model include incremental clustering, and semi-supervised clustering. Other issues that can be further researched include the estimation and statistical basis of the optimum number of clusters and *meta*-features to be used.

References

1. Aas, K., Eikvil, L.: Text Categorization: A Survey. (1999)
2. Baker, L.D., McCallum, A.K.: Distributional clustering of words for text classification. Proceedings of SIGIR98, 21st ACM International Conference on Research and

Development in Information Retrieval, Melbourne, AU, ACM Press, New York, US (1998) 96103
3. Bekkerman, R., El-Yaniv, R., Tishby, N., Winter, Y.: On Feature Distributional Clustering for Text Categorization. Proceedings of SIGIR01, 24th ACM International Conference on Research and Development in Information Retrieval, New Orleans, US, ACM Press, New York, US (2001) 146153
4. Bekkerman, R., El-Yaniv, R., Winter, Y.: Distributional Word Clusters vs. Words for Text Categorization. Journal of Machine Learning Research 3 (2003) 1183-1208.
5. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. Proceedings of the 11th Annual Conference on Computational Learning Theory, (1998) 92-100, ACM Press, NY
6. Dhillon, I., Mallela, S., Kumar, R.: Enhanced word clustering for hierarchical text classification. Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, (2002) 191-200
7. Dhillon, I., Mallela, S., Kumar, R.: A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification. Journal of Machine Learning Research 3 (2003) 1265-1287
8. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of the 10th European Conference on Machine Learning (ECML) (1998)
9. Joachims, T.: Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, (1999)
10. Joachims, T.: Transductive inference for text classification using support vector machines. Proceedings of 16th International Conference on Machine Learning. San Francisco: Morgan Kaufmann (1999) 200-209
11. Karypis, G.: CLUTO a clustering toolkit. Technical Report 02-017, Dept. of Computer Science, University of Minnesota (2002) Available at http://www.cs.umn.edu/cluto.
12. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of ECML-98, 10th European Conference on Machine Learning, pages 415, Chemnitz, DE (1998)
13. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning (1999) 1-34
14. Ng, H.T., Goh, W.B., Low, K.L.: Feature selection, perceptron learning, and a usability case study for text categorization. Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval, Philadelphia, US (1997) 6773
15. Pereira, F., Tishby, N., Lee, L.: Distributional clustering of English words. Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (1993) 183-190
16. Raskutti, B., Ferr, H., Kowalczyk, A.: Using unlabeled data for text classification through addition of cluster parameters. Proceedings of the 19th International Conference on Machine Learning ICML (2002)
17. Raskutti, B., Ferr, H., Kowalczyk, A.: Combining clustering and co-training to enhance text classification using unlabeled data. Proceedings of SIGKDD 02, Canada (2002)
18. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. New York: McGraw-Hill (1983)

19. Schohn, G., Cohn, D.: Less is More: Active Learning with Support Vector Machines. Proceedings of ICML-00, 17th International Conference on Machine Learning (2000)
20. Sebastiani, F.: A tutorial on automated text categorization. Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence (1999) 7-35
21. Slonim, N., Tishby, N.: The power of word clustering for text classification. Proceedings of the European Colloquium on IR Research, ECIR (2001)
22. Takamura, H., Matsumoto, Y.: Two-dimensional Clustering for Text Categorization. Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002), Taipei, Taiwan, (2002) 29-35
23. Takamura, H.: Clustering approaches to text categorization. Doctor's thesis (2003)
24. Tishby, N., Pereira, F.C., Bialek, W.: The Information Bottleneck Method. Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing (1999)
25. Tong, S., Koller, D.: Support Vector Machine Active Learning with Applications to Text Classification. Proceedings of ICML-00, 17th International Conference on Machine Learning (2000)
26. Vapnik, V.: The nature of statistical learning theory. Springer, NY (1995)
27. Weiss, S.M., Apte, C., Damerau, F.J., Johnson, D., Oles, F.J., Goetz, T., Hampp, T.: Maximizing text-mining performance. Intelligent Information Retrieval, IEEE (1999)
28. Wiener, E., Pedersen, J.O., Weigend, A.S.: A neural network approach to topic spotting. Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US (1995) 317332
29. Yang, Y.: Expert network: an effective and efficient learning from human decisions in text categorization and retrieval. Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, Dublin, IE (1994) 13-22
30. Yang, Y., Chute, C.G.: An example-based mapping method for text categorization and retrieval. ACM Transactions of Information Systems, 12(3) (1994) 252-277
31. Zeng, H.J., Wang, X.H., Chen, Z., Lu, H., Ma, W.Y.: CBC: Clustering based text classification requiring minimal labeled data. Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, Florida, USA (2003)
32. Zhao, Y., Karypis, G.: Topic-driven Clustering for Document Datasets. SIAM Data Mining Conference (2005)