A Two-Stage Algorithm for Domain Adaptation with Application to Sentiment Transfer Problems

Qiong Wu[1,2], Songbo Tan[1], Miyi Duan[1] and Xueqi Cheng[1]

[1]Institute of Computing Technology, Chinese Academy of Sciences, China
[2] Graduate University of Chinese Academy of Sciences, China
{wuqiong,tansongbo,duanmiyi}@software.ict.ac.cn, cxq@ict.ac.cn

Abstract. Classification systems are typically domain-specific, and the performance decreases sharply when transferred from one domain to another domain. Building these systems involves annotating a large amount of data for every domain, which needs much human labor. So, a reasonable way is to utilize labeled data in one existing (or called source) domain for classification in target domain. To address this problem, we propose a two-stage algorithm for domain adaptation. At the first transition stage, we share the information between the source domain and the target domain to get some most confidently labeled documents in the target domain, and at the second transmission stage, we exploit them to label the target-domain data via following the intrinsic structure revealed by the target domain. The experimental results on sentiment data indicate that the proposed approach could improve the performance of domain adaptation dramatically.

Keywords: Domain Adaptation, Sentiment Classification, Information Retrieval.

1 Introduction

Text classification [7, 16] is a critical technique in the Natural Language Processing (NLP) community. Many research tasks rely on accurate classification. For example, sentiment classification [4][10] is essentially a text classification problem, and it classifies documents into positive or negative category.

In most cases, a variety of supervised classification methods can perform well. But when training data and test data are from different domains, the supervised classification methods often cannot perform well [8,11,12]. The reason is that training data do not have the same distribution as test data, so test data could not share the information from training data. Therefore, the labeled data in the same domain with test data is considered as the most valuable resources for classification. However, such resources in different domains are very imbalanced. In some traditional domains or domains of concern, many labeled data are freely available on the web, but in other domains, labeled data are scarce and it involves much human labor to manually label reliable data. So, the challenge is how to utilize labeled data in one domain (that is, source domain) for classification in another domain (that is, target domain). This

raises a fundamental and important task, domain adaptation.

Realizing the challenges posed by domain adaptation, some researchers have explored a number of techniques to improve the performance of domain adaptation. However, the difficulties for domain adaptation are as follows: the first one is that the distribution of the target domain is not same with that of the source domain, and hence we need to share the information got from the source domain; the second one is the intrinsic structure of the target domain is static, so we need to utilize the intrinsic structure collectively revealed by target domain. In brief, a good method for domain adaptation is expected to utilize the information contained in the source domain as much as possible, and moreover, follow the intrinsic structure revealed by target domain as much as possible.

In light of the difficulties for domain adaptation, we propose a novel system algorithm which is composed of two stages. The first stage is called transition stage, where we firstly share information between the source and target domains by applying the adapted Prototype algorithm to label the target-domain documents initially, and then choose some high-quality seeds from the target domain which are most confidently labeled. The second stage is called transmission stage, where we utilize the intrinsic structure of the target domain by employing the manifold-ranking process to compute the manifold-ranking score for every document that denotes the degree of belonging to a category. So we can label the target-domain data based on these scores.

Our contribution is as follows. First, while existing domain-adaptation approaches typically rely on a generative model, our approach associates two domains to get some high-quality seeds from the target domain, and then follows the structure embodied by the seeds to improve the performance of domain adaptation. Second, while existing manifold-ranking-based approaches typically start with manually labeled seeds, our approach relies only on seeds that are automatically extracted and labeled from the target domain.

For evaluation, we use the proposed algorithm for sentiment classification problem, and evaluate it on three domain-specific sentiment data sets. The experiment results show that our approach can dramatically improve the accuracy when transferred to another target domain.

2 A Two-Stage Algorithm for Domain Adaptation

2.1 The Transition Stage

2.1.1 Overview

At this stage, we take into account these two objectives: we aim to (1) get the labels of the target-domain documents while utilizing the information of the source domain, and then (2) identify the high-quality documents which are most confidently labeled.

As we know, the features of source domain can be divided into two categories: domain-specific and nondomain-specific, and only nondomain-specific features can be used as a bridge between the source domain and the target domain. That is, we can

use nondomain-specific features to train a classifier for target domain. For the sake of convenience, we call nondomain-specific features as "generalization features". As mentioned in Section 1, traditional Prototype classifier requires both source domain and target domain have the same distribution, which couldn't be satisfied in domain-adaptation problems. So we use an adapted Prototype classifier to solve this problem. An intuitive description is as follows: We pick up generalization features utilizing Frequently Co-occurring Entropy (FCE) [11], and then we only use these features in both source domain and target domain. After that, we assign a score for every unlabeled document to denote its extent to each category, and then the score is calculated by making use of a Prototype classifier trained by the accurate labels of source-domain data. The final score for classification is achieved when the algorithm ends, so the target-domain data can be labeled based on these scores.

2.1.2 Frequently Co-occurring Entropy

In order to pick out generalization features, we use Frequently Co-occurring Entropy (for simplicity, we call it as "f-score" in this work) proposed in [11]. The criteria behind FCE are: (1) occur frequently in both domains; (2) has similar occurring probability. The formula that satisfies these criteria is as follows:

$$f_w = \log\left(\frac{P_S(w)\cdot P_T(w)}{|\,P_S(w)-P_T(w)\,|}\right) \tag{1}$$

where $P_S(w)$ and $P_T(w)$ denote the probability of word w in the source domain and the target domain respectively:

$$P_S(w) = \frac{(N_w^S + \alpha)}{(|\,D^S\,| + 2\cdot\alpha)} \tag{2}$$

$$P_T(w) = \frac{(N_w^T + \alpha)}{(|\,D^T\,| + 2\cdot\alpha)} \tag{3}$$

where N^S_w and N^T_w is the number of examples with word w in the source domain and the target domain respectively; $|D^S|$ and $|D^T|$ is the number of examples in the source domain and the target domain respectively. And according to [11], α is set to 0.00001.

After computing the f-score of every word, we rank the words in descending order according to their f-scores. Then, we choose the first $K_{FEATURE}$ documents as generalization features.

In formula (1), "$P_S(w)$ $P_T(w)$" embodies the first criterion, and "$|P_S(w)$-$P_T(w)|$" embodies the second criterion. For the sake of being easy to adjust the weight of the two factors, a trade-off parameter $\boldsymbol{\pi}$ is introduced in the formula. Meanwhile, in case of $P_S(w) = P_T(w)$, a parameter $\boldsymbol{\beta}$ is introduced. Therefore, the formula is modified as follows:

$$f_w = \log\left(\frac{(P_S(w)\cdot P_T(w))^{\pi}}{(|P_S(w)-P_T(w)|+\beta)^{(1-\pi)}}\right) \quad (4)$$

where β is set as 0.0001 according to [11].

2.1.3 Prototype Classifier

Since we have got generalization features, we process both source-domain and target-domain data so that they only contain generalization features. After that, we assign every unlabeled document a score to represent its degree of belonging to a category, and the score is a real number between -1 and 1. Here, we assume that we classify the documents into two categories: category 1, and category 2. Therefore, when the score is between 0 and -1, the document should be classified as "category 1". The closer its score is near -1, the higher the "category 1" degree is, when the score is between 0 and 1, the document should be classified as "category 2". The closer its score is near 1, the higher the "category 2" degree is. We use S^{Target} to denote the score set of target domain.

In order to compute the score vector S^{Target}, we need to use a traditional classifier. There are many kinds of classifiers. For simplicity, we take prototype classification algorithm [7] as an example here. And its process in our context can be described as follows:

1. Compute the center vector C_1 and C_2 for document set belonging to category 1 (D_1^{SOURCE}) and document set belonging to category 2 (D_2^{SOURCE}) of source domain as follows:

$$C_l = \sum_{i\in D_l^L} d_i \Big/ \left|D_l^{SOURCE}\right|, \quad l \in \{1, 2\} \quad (5)$$

where |.| is the cardinality of a set.

2. Calculate the similarity between $d_i \in$ target domain (i = 1,…n) and each center vector with the cosine measure as follows:

$$Sim_l = \frac{d_i \cdot C_l}{\|d_i\| \times \|C_l\|}, \quad l \in \{1, 2\} \quad (6)$$

3. Assign $d_i \in$ target domain (i= 1,…n) a class label corresponding to the more similar center vector.
4. Give the opposite number of similarity value with C_1 to s_i if d_i's label is "category 1", e.g. if d_i's label is "category 1", and its similarity value with C_1 is 0.5, then set s_i to -0.5. Give the similarity value with C_2 to s_i if d_i's label is "category 2", e.g. if d_i's label is "category 2", and its similarity value with C_2 is 0.5, then set s_i to 0.5.

2.1.4 Identify the High-Quality Documents

Next, we find the high-quality documents from the target domain. To do this, we make use of the score which denotes its corresponding document's extent to "category 1" or "category 2". Firstly, we rank the target-domain documents in descending order according to their scores. So the more forward the document is ranked, the more likely it belongs to category 2; the more backward the document is ranked, the more likely it belongs to category 1. Then, we choose the first K documents and last K documents as the high-quality documents. In the rest of the paper, we will refer to these high-quality documents as seeds.

2.2 The Transmission Stage

The adapted Prototype classifier allows us to share information between the source domain and the target domain, but we haven't utilized the distribution of the target domain. In fact, being able to follow the intrinsic structure of the target domain is important for domain adaptation, as discussed before. Now that we have a small, high-quality seed set which embodies the intrinsic structure of the target domain, we can make better use of the seeds by utilizing the manifold-ranking method and having it improve the performance of domain adaptation.

The manifold-ranking method [13] is a universal ranking algorithm and it is initially used to rank data points along their underlying manifold structure. The prior assumption of manifold-ranking is: (1) nearby points are likely to have the same ranking scores; (2) points on the same structure (typically referred to as a cluster or a manifold) are likely to have the same ranking scores. An intuitive description of manifold-ranking is as follows: a weighted network is formed on the data, and a positive rank score is assigned to each known relevant point and zero to the remaining points which are to be ranked. All points then spread their ranking score to their nearby neighbors via the weighted network. The spread process is repeated until a global stable state is achieved, and all points obtain their final ranking scores.

Given that we now have a high-quality seed set, we build the weighted network whose points denote documents in target domain. Also, we integrate the scores of the seeds into the manifold-ranking process. So the manifold-ranking process used for domain adaptation can be formalized as follows:

Given a point set $\chi = \{x_1,\ldots,x_K,x_{K+1},\ldots,x_{2K}, x_{2K+1},\ldots,x_n\} \subset R^m$, the first K points x_i ($1 \le i \le K$) are the seeds which are labeled "category 2", the second K points x_j ($K+1 \le j \le 2K$) are the seeds which are labeled "category 1", and the remaining points x_u ($2K+1 \le u \le n$) are unlabeled. Let $F : \chi \to R^2$ denote a ranking function which assigns to each point x_i ($1 \le i \le n$) a ranking value vector F_i. We can view F as a matrix $F = [F_1^T,\ldots,F_n^T]^T$. We also define a $n \times 2$ matrix $Y = [Y_1,Y_2]$, where $Y_1 = [Y_{11},\ldots,Y_{K1},Y_{K+1,1},\ldots,Y_{n1}]^T$ and $Y_2 = [Y_{12},\ldots,Y_{K2},Y_{K+1,2},\ldots,Y_{n2}]^T$ with $Y_{i1}=1$ if x_i is labeled as "category 2" and $Y_{i2}=1$ is x_i is labeled as "category 1". The manifold ranking algorithm used for domain adaptation goes as follows:

1. Compute the pair-wise similarity values between points using the cosine measure. The weight associated with term t is calculated with the tf_t*idf_t formula, where tf_t

is the frequency of term t in the document and idf_t is the inverse document frequency of term t, i.e. $1+\log(N/n_t)$, where N is the total number of documents and n_t is the number of the documents containing term t. Given two points x_i and x_j, the cosine similarity is denoted as $sim(x_i,x_j)$, computed as the normalized inner product of the corresponding term vectors.

2. Connect any two points with an edge if their similarity isn't 0. We form the affinity matrix W defined by $W_{ij}=sim(x_i,x_j)$ if $i \neq j$, and we let $W_{ii}=0$ to avoid loops in the graph built in the next step.
3. Construct the matrix $S=D^{-1/2}WD^{-1/2}$ in which D is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of W.
4. Iterate $F(t+1)=\alpha SF(t)+(1-\alpha)Y$ until convergence, where α is a parameter in (0,1).
5. Let F^* denote the limit of the sequence $\{F(t)\}$. Then every document x_j $(K+1 \leq j \leq n)$ gets its ranking score vector F_j^*.

In the manifold-ranking algorithm, the weight matrix W is normalized symmetrically in the third step, which is necessary to prove the algorithm's convergence. During the fourth step, every point receives the information from its neighbors (first term), and also retains its initial information (second term). The parameter of manifold-ranking weight α specifies the relative contributions to the ranking scores from its neighbors and its initial ranking scores. According to [13], in our experiment, we set α to 0.6. It is worth mentioning that self-reinforcement is avoided since the diagonal elements of the affinity matrix are set to zero in the second step. Moreover, the information is spread symmetrically since S is a symmetric matrix.

Zhou et al. [13] proves that the sequence $\{F(t)\}$ converges to

$$F^* = \beta(I - \alpha S)^{-1}Y \tag{7}$$

In the above formula, $\beta = 1 - \alpha$. Note that although F^* can be expressed in a closed form, for large scale problems, the iteration algorithm is preferable due to computational efficiency. Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any point falls below a given threshold (0.00001 in this study).

Finally, we label the documents in target domain according to their ranking score vector. For each document x_i $(1 \leq i \leq n)$, if $Y_{i1} > Y_{i2}$, assign the document the label "category 2"; if $Y_{i1} < Y_{i2}$, assign the document the label "category 1".

3 Evaluation

We evaluate our two-stage algorithm in the field of sentiment classification. In order to highlight the domain-specific nature of sentiment expression, we use reviews not only from different web sites, but also from domains with less similarity. Aimed at Chinese applications, we conduct the experiments based on the characteristics of the Chinese, and verify the performance on Chinese web reviews. However, the main proposed approach in this paper is language independent in essence.

3.1 Experimental Setup

For evaluation, we use three Chinese domain-specific data sets from on-line reviews, which are: Book Reviews[1] (B, from http://www.dangdang.com/), Hotel Reviews[2] (H, from http://www.ctrip.com/) and Notebook Reviews [3] (N, from http://www.360buy.com/). Each dataset has 4000 labeled reviews (2000 positives and 2000 negatives). We choose one of the three data sets as source-domain data, and another data set as target-domain data.

3.2 Baseline Systems

In this paper we compare our approach with the following baseline methods:

Proto: This method applies a traditional supervised classifier, Prototype classifier [7], for the sentiment transfer. And it only uses source domain documents as training data. Results of this baseline are shown in column 1 of Table 1. As we can see, accuracy ranges from 61.25% to 73.5%.

TSVM: This method applies transductive SVM for the sentiment transfer which is a widely used method for improving the classification accuracy. In our experiment, we use Joachims's SVM-light package (http://svmlight.joachims.org/) for TSVM. We use a linear kernel and set all parameters as default. This method uses both source domain data and target domain data, obtaining the results in column 2 of Table 1. As we can see, accuracy ranges from 61.42% to 77.17%, which are better than Proto.

EM: We implement the EM algorithm [6] based on Prototype classifier. Specifically, we train the Prototype classifier on the data labeled so far, use it to get the sentiment scores of unlabeled documents in the target domain, and augment the labeled data with K_E most confidently labeled documents. We test values for K_E from 10 to 300, an increase of 20 each, and reported in column 4 of Table 1 the best results. As we can see, since EM is based on Prototype, its accuracy ranges from 65.7% to 76.5% which are better than other baselines except Manifold.

Manifold: Our last baseline implements the manifold-ranking procedure [13] adaptable for sentiment transfer. Specifically, we begin by training a prototype classifier on the training data. Then we use the similarity scores between the documents and the positive central vector and the similarity scores between the documents and the negative central vector to separately initialize the ranking score vectors of the test data. Finally, we choose K_M documents that are most likely to be positive and K_M documents that are most likely to be negative as seeds for manifold-ranking. We test values for K_M from 10 to 300, an increase of 20 each, and reported in column 5 of Table 1 the best results. As we can see, accuracy ranges from 66.5% to 78.4% which are better than all other baselines.

---

1 http://www.searchforum.org.cn/tansongbo/corpus/Dangdang_Book_4000.rar.

2 http://www.searchforum.org.cn/tansongbo/corpus/Ctrip_htl_4000.rar.

3 http://www.searchforum.org.cn/tansongbo/corpus/Jingdong_NB_4000.rar.

Table 1. Accuracy comparison of different methods

Domain	Proto	TSVM	Adapted Proto	EM based on Proto	Manifold based on Proto	Our Approach
B->H	0.735	0.749	0.751	0.765	0.761	**0.766**
B->N	0.651	0.769	0.678	0.667	0.745	**0.748**
H -> B	0.645	0.614	0.670	**0.723**	0.677	0.695
H ->N	0.729	0.726	0.744	0.657	**0.784**	0.775
N-> B	0.612	0.622	0.653	**0.763**	0.665	0.702
N-> H	0.724	0.772	0.783	0.765	0.779	**0.806**
Aver	0.683	0.709	0.713	0.723	0.735	**0.748**

3.3 Our Approach

In this section, we compare the adapted Prototype classifier (Adapted Proto in Table 1) and the two-stage approach (Our Approach in Table 1) with three baseline methods. There is a parameter, K, in our two-stage approach. We set K to 290 to show we choose $2K$ seeds for manifold-ranking algorithm.

Results of the adapted Prototype classifier are shown in column 3 of Table 1. As we can observe, the adapted Prototype classifier produces much better performance than Proto. The greatest average increase of accuracy is achieved by about 3% compared to Proto. The great improvement indicates that the adapted Prototype classifier is more effective to share information between source domain and target domain.

Results of our approach are shown in column 6 of Table 1. As we can observe, our approach produces much better performance than all the baselines. The greatest average increase of accuracy is achieved by about 6.5% compared to Proto. The great improvement compared with the baselines indicates that our approach performs very effectively and robustly.

Table 1 shows the average accuracies of Adapted Proto and TSVM are higher than Proto: the average accuracy of Adapted Proto is about 3% higher than Proto, and the average accuracy of TSVM is about 2.6% higher than Proto. As we know, Adapted Proto and TSVM utilize information of both source domain and target domain while Proto not, so this proves that utilizing the information of two domains is better than utilizing the information of only one domain for improving the accuracy of sentiment transfer.

Seen from Table 1, the average accuracies of the last three columns are higher than the first three columns. As we know, the last three approaches are all two-stage approaches while the first three approaches are not, so this indicates that two-stage transfer approach is more effective for sentiment transfer.

Meanwhile, the average accuracy of Manifold based on Adapted Proto showed in column 6 is higher than Manifold based on Proto showed in column 5: the average increase of accuracy of Manifold based on Adapted Proto is achieved by about 1.3%

compared to Manifold based on Proto, which proves that Adapted Proto can choose higher quality seeds that embody the intrinsic structure of the domain for next stage.

Table 1 shows the Manifold based on Proto approach outperforms EM: the average accuracy of Manifold based on Proto is about 1.3% higher than EM based on Proto, and the greatest increase of accuracy is achieved by about 13% on the task "H->N" compared to EM based on Proto. This is caused by two reasons. First, EM is not dedicated for sentiment-transfer learning. Second, the manifold approach can follow the intrinsic structure of the target domain better.

4 Related Work

4.1 Classification

Classification is a traditional and widely-used problem, which includes two steps. The first step is to establish a classifier which describes the predefined training data. This is called training stage. The second step is using the classifier to classify test data. This is called test stage. It generally includes supervised classification and semi-supervised classification. Supervised classification is used when there are a lot of labeled data. Effective methods include Naïve Bayes classifiers [17]、KNN [18]、decision tree [19] 、neural network [20] and support vector machines [21] et al. Semi-supervised classification (e.g. [22]~[24]) is used when labeled data are not sufficient to build a classifier. It uses both a small amount of labeled data and a large amount of unlabeled data to build a classifier. Nigam [22] used EM algorithm [6] to get pseudo labels of unlabeled data, and then incorporate them into Naïve Bayes classifier. Lanquillon [23] put forward a framework to utilize unlabeled data using an Expectation-Maximization-like scheme. Joachims [24] exploited the unlabeled date to modify SVM.

Typical classification requires the labeled and unlabeled data should be under the same distribution. So the classifier built by the labeled data could be well applied to the unlabeled data. But in our domain adaptation problem, the labeled and unlabeled data are from different domains, and often have different distributions. This is inconsistent with the basic requirement of typical classification, and the effect methods cannot be directly used in our problem.

4.2 Domain Adaptation

Domain adaptation aims to utilize labeled data from other domains or time periods to help current learning task, and the underlying distributions are often different from each other.

In the past years, many researchers have been working on this field and have proposed many approaches, including classifier adaptation [5], two-stage approach [8], consensus regularization framework [9] and so on. DaumeIII and Marcu [14] studied the domain-transfer field in statistical natural language processing using a specific Gaussian model. Then they presented an instantiation of this framework to Maximum

Entropy classifiers and their linear chain counterparts. Xing et al. [15] proposed a novel algorithm, namely bridged refinement. They took the mixture distribution of the training and test data as a bridge to better transfer from the training data to the test data. Jiang and Zhai [8] presented a two-stage approach for transfer learning. At the first generalization stage, they got a set of features generalizable across domains, and at the second adaptation stage, they picked up useful features specific to the target domain. Then they also proposed a number of heuristics to approximately achieve the goal of generalization and adaptation.

Recently, some researchers attempted to address sentiment domain adaptation [3][12].

Some studies rely on only the labeled documents to improve the performance of sentiment transfer (e.g. [2]; [11]). Aue and Gamon [2] used four different approaches to customize a sentiment classification system to a new target domain using a small amount of labeled training data. Tan et al. [11] proposed Frequently Co-occurring Entropy to pick out generalizable features that occurred similarly in both domains, and then proposed Adapted Naïve Bayes to train a classifier suitable for the target-domain data.

Moreover, some studies rely on only the sentiment words to improve the performance of sentiment transfer (e.g. [1]). Andreevskaia and Bergler [1] presented a sentiment annotation system that integrated a corpus-based classifier trained on a small set of annotated in-domain data and a lexicon-based classifier trained on WordNet.

However, most of the existing studies rely on only utilizing the information from the source domain to address the task of domain adaptation, while ignoring the intrinsic structure of the target domain.

In this paper, we design an algorithm for domain adaptation by taking into account the relationship between source domain and target domain as well as the intrinsic structure of the target domain.

5 Conclusions

We propose a two-stage algorithm for domain adaptation. Our key idea is to share information between the source domain and the target domain, and follow the intrinsic structure of the target domain to improve the performance of domain adaptation. Specifically, the algorithm consists of a transition stage and a transmission stage. At the transition stage, we (1) share the information between the source domain and the target domain with the help of an adapted Prototype classifier to get the scores of the target-domain documents, (2) utilize the scores to identify a small number of most confidently labeled documents as high-quality seeds. At the transmission stage, we (3) employ the manifold-ranking algorithm to spread the seeds' ranking scores to their nearby neighbors to compute the ranking score for every unlabeled document, (4) label the target-domain data based on these scores.

Experimental results on three domain-specific sentiment data sets demonstrate that our approach can dramatically improve the accuracy, and can be employed as a high-performance domain adaptation system.

In future work, we plan to evaluate our algorithm to many more tasks. Also, we plan to try more algorithms for each stage of our algorithm.

6 Acknowledgments

This work was mainly supported by two funds, i.e., 60933005 & 60803085, and two other projects, i.e., 2007CB311100 & 2007AA01Z441.

References

1. Andreevskaia, A., Bergler, S.: When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. In: ACL 2008, pp. 290-298 (2008)
2. Aue, A., Gamon, M.: Customizing sentiment classifiers to new domains: a case study. In: RANLP 2005 (2005)
3. Tan, S., Wang, Y., Wu, G., Cheng, X.: A novel scheme for domain-transfer problem in the context of sentiment analysis. In: CIKM 2007, pp. 979-982 (2007)
4. Cui, H., Mittal, V., Datar, M.: Comparative experiments on sentiment classification for online product reviews. In: AAAI 2006, pp. 1265-1270 (2006)
5. Dai, W., Xue, G., Yang, Q., Yu, Y.: Transferring Naive Bayes Classifiers for Text Classification. In: AAAI 2007, pp. 540-545 (2007)
6. Dempster, A.P., Laird, N.M., Rubin D.B.: Maximum Likelihood from Incomplete Data via the EM algorithm. Journal of the Royal Statistical Society, 39(B):1-38 (1977)
7. Tan, S., Cheng, X., Ghanem, M., Wang, B., Xu, H.: A Novel Refinement Approach for Text Categorization. In: CIKM 2005, pp. 469-476 (2005)
8. Jiang, J., Zhai, C.X.: A Two-Stage Approach to Domain Adaptation for Statistical Classifiers. In: CIKM 2007, pp. 401-410 (2007)
9. Luo, P., Zhuang, F., Xiong, H., Xiong, Y., He, Q.: Transfer learning from multiple source domains via consensus regularization. In: CIKM 2008, pp. 103-112 (2008)
10. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: EMNLP 2002, pp. 79-86 (2002)
11. Tan, S., Cheng, X., Wang, Y., Xu, H.: Adapting Naïve Bayes to Domain Adaptation for Sentiment Analysis. In: ECIR 2009, pp. 337-349 (2009)
12. Wu, Q., Tan, S., Cheng, X.: Graph Ranking for Sentiment Transfer. In: ACL 2009, pp. 317-320 (2009)
13. Zhou, D., Weston, J., Gretton, A., Bousquet O., Schölkopf, B.: Ranking on data manifolds. In: NIPS 2003, pp. 169-176 (2003)
14. DaumeIII, H., Marcu, D.: Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research, 2006, 26: 101-126 (2006)
15. Xing, D., Dai, W., Xue, G., Yu Y.: Bridged refinement for transfer learning. In: PKDD 2007, pp. 324-335 (2007)
16. Tan, S.: An Effective Refinement Strategy for KNN Text Classifier. Expert Systems With Applications. Elsevier. 2006, 30(2): 290-298 (2006)
17. Lewis, D.: Representation and Learning in Information Retrieval. PhD thesis, Amherst, MA, USA (1992)
18. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 1967, 13 :21-27 (1967)

19. Quinlan, J.R.: Induction of decision trees. Machine Learning , 1986, 1 :81-106 (1986)
20. Carrol, S.M., Dickinson, W.: Construction of Neural Nets Using Radom Transform. In: Proc. IJCNN,1989,(1):607-611 (1989)
21. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: ICML 1998, pp. 137-142 (1998)
22. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, 39(2-3):103-134, 2000 (2000)
23. Lanquillon, C.: Learning from Labeled and Unlabeled Documents: A Comparative Study on Semi-Supervised Text Classification. In: PKDD 2000, pp. 167-194 (2000)
24. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML 1999, pp. 200-209 (1999)