Semi-supervised Learning with Transfer Learning

Huiwei Zhou, Yan Zhang, Degen Huang, and Lishuang Li

Dalian University of Technology, Dalian, Liaoning, China
{zhouhuiwei,huangdg}@dlut.edu.cn, 358742700@mail.dlut.edu.cn,
lilishuang314@163.com

Abstract. Traditional machine learning works well under the assumption that the training data and test data are in the same distribution. However, in many real-world applications, this assumption does not hold. The research of knowledge transfer has received considerable interest recently in Natural Language Processing to improve the domain adaptation of machine learning. In this paper, we present a novel transfer learning framework called TPTSVM (Transfer Progressive Transductive Support Vector Machine), which combines transfer learning and semi-supervised learning. TPTSVM makes use of the limited labeled data in target domain to leverage a large amount of labeled data in source domain and queries the most confident instances in target domain. Experiments on two data sets show that TPTSVM algorithm always improves the classification performance compared to other state-of-the-art transfer learning approaches or semi-supervised approaches. Furthermore, our algorithm could be extended to multiple source domains easily.

1 Introduction

Traditional machine learning assumes that the training data and test data are in the same distribution and the training data is sufficient to get an acceptable model [1][2][3]. But in many real works, the training data are always scarce and it is expensive to label the sufficient training data. For example, in Web-document classification, the labeled data used for training may be easily outdated or under a different distribution from the new data [4][5][6]. In such cases, it would be helpful if we could use unlabeled new data or transfer the classification knowledge into the new domain.

Transfer learning is a machine learning algorithm that allows the distributions, domains, even tasks used in training data and testing data differently [7][8]. Recently, research on transfer learning has attracted more and more interest in several topics, such as knowledge transfer [9], learning to learn [10], multi-task learning [11], domain transfer [12]. Among these, we address on the domain transfer problem, which aims to transfer classification knowledge from the source domain to the target domain in the same task.

Domain-transfer learning has been studied and works well when the distributions of source and target domain are similar [13][14]. But significant distribution divergence might cause negative transfer [15], which is the limitation of transfer learning. Finding the inner relationship between source domain and target domain and using the source data as far as possible are the problems must be solved. TrAdaBoost [16]

M. Sun et al. (Eds.): CCL and NLP-NABD 2013, LNAI 8202, pp. 109–119, 2013.

decrease the negative effects of source domain and boost the accuracy on target domain by Boosting. TransferBoost [17] adjusts the weights of each source domain according to its positive transferability to the target domain. Active Vector Rotation (AVR) [18] uses active learning to avoid negative transfer and reduces the labeling cost. Most current transfer learning researches focus on using a large set of labeled source data and a small set of labeled target data [19][20]. However, the limited labeled target data can not represent the whole target domain sufficiently.

On the other hand, semi-supervised learning is another effective machine learning strategy when the training sets are small, for example, in sentiment analysis [21]. Including a particular set of unlabeled working or testing data, semi-supervised learning can be used to improve the generalization performance for both training and working data set. Transductive Support Vector Machines (TSVM) [22] is a well known semi-supervised learning algorithm, which achieves better performance than traditional inductive SVM, especially for small training sets. However, TSVM assumes that the training and working data follow the same ratio of positive/negative examples. To handle different class distribution, progressive transductive support vector machine (PTSVM) [23] labels and modifies the unlabeled working data set with the rule of pairwise labeling and dynamical adjusting. Semi-supervised learning only uses a small number of labeled training data and a large number of unlabeled testing data. A large number of labeled data from a similar old domain often provide some useful information. Throwing away all old data would also result in a waste.

In this paper, we propose a framework, named transfer progressive transductive support vector machine (TPTSVM), which takes advantage of both transfer learning and semi-supervised learning techniques. TPTSVM tries to make use of the limited labeled data in target domain to leverage a large amount of labeled data in source domain and queries the most confident instances in target domain. The weights of individual instances are carefully adjusted on both the instance level and the domain level. In our experiments, we only use one source domain, but our algorithm could be extended to multiple source domains easily. The experiments show that semi-supervised learning can improve the transfer learning's performance and our algorithm works well especially for insufficient training set.

2 Transfer Progressive Transductive Support Vector Machine

To simplify the question, we constrain the discussion of TPTSVM to binary classification tasks and transfer knowledge from one source domain to the target domain. But in fact, TPTSVM can also be extended to multi-class and multi-source. Given three data sets, the target domain labeled data set $D_l = \{(x_i^l, y_i^l) \mid i = 1, ..., n\}$, $y_i^l = \{-1, +1\}$, the target domain unlabeled data set $D_u = \{(x_j^u) \mid j = 1, ..., m\}$, and the source domain labeled data set $D_s = \{(x_k^s, y_k^s) \mid k = 1, ..., r\}$, $y_k^s = \{-1, +1\}$. n, m and r are the sizes of D_l, D_u and D_s. Assume that D_l and D_u are in the same domain with same distribution, but D_l is not sufficient to training a model to classify the D_u. Our TPTSVM algorithm tries to use both D_s and D_u to help the unsufficient training set D_l to train a better

Algorithm 1

Input the two labeled data sets D_s and D_l, the unlabeled data set D_u, a SVM learner SVM($D(x_i)$,$W(x_i)$), the number of iteration N, the growth size p at each iteration (p/2 positive and p/2 negative confident instances).

Initialize

a) $D_{ul} = \{\}$, $D_{uu} = D_u$.

b) $W^1 = (w(x_i)) = 1,\ x_i \in D_s \bigcup D_l \bigcup D_{ul}$,

$W_t \in W = (w(x_i)),\ x_i \in D_l \bigcup D_{ul}$

c) $\beta = \dfrac{1}{1+\sqrt{2\ln|D_s|/N}}$

For d=1,2,...,N

1. Call learner SVM($D_s \bigcup D_l \bigcup D_{ul}, W^d$), get back a hypothesis function $f(x_i)$;

   Call learner SVM($D_l \bigcup D_{ul}, W_t^d$), get back a hypothesis function $f_t(x_i)$.

2. Calculate the error of $f(x_i)$ on D_l:

$$\varepsilon = \sum_{x_i \in D_l} \frac{W_t(x_i) * |sign(f(x_i)) - y_i|}{2 * \sum_{x_i \in D_l} W_l^d(x_i)}$$

   Calculate the error of $f(x_i)$ on $D_l \bigcup D_{ul}$:

$$\varepsilon_s = \sum_{x_i \in D_l \bigcup D_{ul}} \frac{W_t(x_i) * |sign(f(x_i)) - y_i|}{2 * \sum_{x_i \in D_l \bigcup D_{ul}} W_t^d(x_i)}$$

   Calculate the error of $f_t(x_i)$ on $D_l \bigcup D_{ul}$:

$$\varepsilon_t = \sum_{x_i \in D_l \bigcup D_{ul}} \frac{W_t(x_i) * |sign(f_t(x_i)) - y_i|}{2 * \sum_{x_i \in D_l \bigcup D_{ul}} W_t^d(x_i)}$$

3. Reweight the instances x_i on D_s:

$$w^{d+1}(x_i) = w^d(x_i) * e^{(\varepsilon_t - \varepsilon_s)} * \beta^{|sign(f(x_i)) - y_i|/2},\ x_i \in D_s;$$

   Reweight the instances x_i on D_l:

$$w^{d+1}(x_i) = w^d(x_i) * (1-\varepsilon)/\varepsilon,\ \text{if } sign(f(x_i)) \neq y_i, x_i \in D_l.$$

4. Calculate the function values $f(x_i)$ of instances x_i on D_{uu} and D_{ul}.
5. Select p/2 positive and p/2 negative the most confident instances from D_{uu} to D_{ul}.
6. Move unconfident instances from D_{ul} back to D_{uu}.
7. Reweight instances x_i on D_{ul}:

$$w^{d+1}(x_i) = \frac{n}{N} * (1-\varepsilon),\ x_i \in D_{ul}.$$

Output the hypothesis function $f(x_i)$.

classifier $f(x_i)$ that minimizes the prediction error on the target domain unlabeled data set D_u. A formal description of the framework is given in Algorithm 1. Where, D_l is not sufficient to train a model to classify the D_u; D_s is the old data set or source domain data set that we try to reuse as much as possible. TPTSVM tries to transfer source instances to learn the target distribution on both the instance level and the domain level. On the domain level, TransferBoost is used to adjust the weight of source domain to overcome the irrelevancies of source domain. On the instance level, the theory of Adaboost is applied to adjust the weights of instances.

Besides D_l and D_s, the learner is also given the unlabeled data D_u from the target domain. D_u is large enough to response the feature and the distribution of the target domain data. We also expect to make use of D_u for target-domain classification. TPTSVM aims to select the most confident instances from D_u into the predicted labeled data set D_{ul} to help learning. Semi-supervised algorithm is a learning framework which aims to label the unlabeled data for training. In our TPTSVM, the improved semi-supervised learning algorithm PTSVM is applied to select the most informative data.

On each iteration round, two models are trained with instances' weights. One uses the union of the source and the target domain data including the predicted labeled data set D_{ul} selected from the unlabeled data set D_u, and the other just uses the training data set in the target domain. TPTSVM trains two classifiers with two sets of training instances together with their weights. It then reweights each instance in the source domain, increasing or decreasing their weights by a factor of $e^{(\varepsilon_t-\varepsilon_s)}$ based on whether the union of the source and target domain data shows positive or negative transfer to the target domain, and reducing their weights by a factor of $\beta^{|sign(f(x_i))-y_i|/2}$ if a source domain instance is mistakenly predicted. The parameter β in algorithm hedge(β) [24] is used to decrease the weight of harmful instances. $e^{(\varepsilon_t-\varepsilon_s)}$ is an influential factor of source domain data. It will be less than 1 when the source domain is irrelevant and larger than 1 when relevant, which makes TPTSVM easy to extend to multiple source domains. To boost the accuracy on the target training data D_l, TPTSVM increases the weights of mispredicted instances using the error rate ε computed from D_l.

Next, TPTSVM selects p/2 positive and p/2 negative instances, which are the most confidently predicted and useful (the hypothesis function value $0.5<|f(x_i)|<1$), from D_{uu} to D_{ul}. Similarly, it moves unconfident instances ($|f(x_i)|<0.5$) from D_{ul} back to D_{uu}. The weights of instances in D_{ul} are increased slightly on each iteration based on the training error ε on the target domain.

After several iterations, the weights of the source domain instances will be changed according to the similarity of distribution between the source data and the target data. In addition, the instances from the source domain that show "good"("bad") effect to the target domain will have higher (lower) training weights, while target instances that are mislabeled will be emphasized. Furthermore, the credible predicted unlabeled target-domain data are available and used to represent the target distribution. In this manner, TPTSVM trains a domain adaptation model from both source and target data via transfer learning and transductive learning.

3 Analysis

In general SVM [22] framework where we cannot separate training instances linearly, by introducing slack variables ξ_i, the primal problem can be written as:

$$\begin{aligned} &\min(\frac{1}{2}\|w\|^2 + C\sum_i \xi_i), \\ &s.t. \qquad y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \end{aligned} \tag{1}$$

It has been proofed that the unlabeled data inside the margin band of the separating hyperplane could improve the performance in semi-supervised machine learning. By including some unlabeled data, TSVM [22] becomes solving the following optimization problem:

$$\begin{aligned} &\text{minimize over } (y_1^*, ..., y_m^*, w, b, \xi_1, ...\xi_n, \xi_1^*, ..., \xi_m^*) \\ &\qquad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + C^*\sum_{j=1}^{m}\xi_j^* \\ &\text{subject to :} \quad \forall_{i=1}^{n} : y_i[w \cdot x_i + b] \geq 1 - \xi_i \\ &\qquad \forall_{j=1}^{m} : y_j^*[w \cdot x_j^* + b] \geq 1 - \xi_j^* \\ &\qquad \forall_{i=1}^{n} : \xi_i \geq 0 \\ &\qquad \forall_{j=1}^{m} : \xi_j^* \geq 0 \end{aligned} \tag{2}$$

Where $y_1^*, ..., y_m^*$ are the predicted labels for the instances in D_u. In our TPTSVM algorithm, not only the target domain unlabeled data, but also the source domain labeled data is added in training set. So the optimization problem becomes:

$$\begin{aligned} &\text{minimize over } (y_1^*, ..., y_m^*, w, b, \xi_1, ...\xi_n, \xi_1^*, ..., \xi_m^*) \\ &\qquad \frac{1}{2}\|w\|^2 + \sum_{k=1}^{r} C_k^{'}\xi_k + \sum_{i=1}^{n} C_i\xi_i + \sum_{j=1}^{m} C_j^*\xi_j^* \\ &\text{subject to :} \quad \forall_{k=1}^{r} : y_k[w \cdot x_k + b] \geq 1 - \xi_k \\ &\qquad \forall_{i=1}^{n} : y_i[w \cdot x_i + b] \geq 1 - \xi_i \\ &\qquad \forall_{j=1}^{m} : y_j^*[w \cdot x_j^* + b] \geq 1 - \xi_j^* \\ &\qquad \forall_{k=1}^{r} : \xi_k \geq 0 \\ &\qquad \forall_{i=1}^{n} : \xi_i \geq 0 \\ &\qquad \forall_{j=1}^{m} : \xi_j^* \geq 0 \end{aligned} \tag{3}$$

The objective function (3) uses three data sets for learning: source domain labeled data, target domain labeled data and target domain unlabeled data. $C_k^{'}$, C_i and C_j^{*} are the loss costs of instances in source domain labeled data, target domain labeled data and target domain unlabeled data respectively. The final optimal separating hyperplane could be found after finite iterations because the loss costs are finite numbers.

To reduce irrelevance of the source domain data, the algorithm Hedge(β) [24] is used in our algorithm. We have the same conclusion with the algorithm Hedge(β):

$$\frac{L_l}{N} \le \min_{1\le j\le r} \frac{L(x_j)}{N} + \sqrt{\frac{2\ln r}{N}} + \frac{\ln r}{N} \tag{4}$$

It indicates that the average training loss (L_l/N) through N iteration on the source domain data D_l is at most $\sqrt{2\ln r/N} + \ln r/N$ larger than the average minimum training loss of the instances ($\min_{1\le j\le r} L(x_j)/N$).

Like algorithm Adaboost [24], our algorithm TPTSVM increases the weights of mispredicted instances in target domain. Therefore, the accuracy on target domain labeled data which is believed as the standard data could be guaranteed.

In algorithm Adaboost, the prediction error ε_p on target domain labeled data satisfies the follow formula:

$$\varepsilon_p \le 2^N \prod_{d=1}^{N} \sqrt{\varepsilon_d (1-\varepsilon_d)} \tag{5}$$

if the final hypothesis is:

$$h_f(x) = \begin{cases} 1 & \text{if} \quad \sum_{d=1}^{N} (\log \frac{1}{\beta_d}) h_d(x) \ge \frac{1}{2} \sum_{d=1}^{N} \log \frac{1}{\beta_d} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

When ε_d is less than 1/2, the prediction error of $h_f(x)$ on the training data becomes increasingly smaller after each iteration.

Formally, if we consider the follow hypothesis

$$h_f^*(x) = \begin{cases} 1 & \text{if} \quad \sum_{d=\tau}^{N} (\log \frac{1}{\beta_d}) h_d(x) = \sum_{d=\tau}^{N} \log \frac{1}{\beta_d} \\ 0 & \text{if} \quad \sum_{d=\tau}^{N} (\log \frac{1}{\beta_d}) h_d(x) = 0 \end{cases} \tag{7}$$

we have:

$$\varepsilon_p \le 2^{N-\tau+1} \prod_{d=\tau}^{N} \varepsilon_d \tag{8}$$

We assume the distribution of the target domain unlabeled data is the same as that of the target domain labeled data. In our algorithm TPTSVM, an unlabeled instance x_τ which is added to D_{ul} after the τ^{th} iteration will still be in D_{ul} after all iteration. x_τ would satisfies the hypothesis $h_f^*(x)$, and the error probability of x_τ would less than $2^{N-\tau+1}\prod_{d=\tau}^{N}\varepsilon_d$. Actually, the error probability of the instances added into D_{ul} earlier will become smaller along with the $N-\tau+1$ growing bigger. On the other hand, the accuracy of the instances added into D_{ul} later is also acceptable, because we increased the weights of mispredicted instances in target domain.

4 Experiment

4.1 Data Sets

The experiments are performed on one none-text data set (mushroom data from the UCI machine learning repository[1]) and one text data set (20 newsgroups data[2]),which are all used in [16]. We also split the data sets to fit our learning problem like [16]. We generate four tasks using the two data sets as shown in Table 1.

The mushroom data contains two categories, edible and poisonous. We split the data based on the stalk-shape feature to make the source and target data have different distribution. The source data set consists of all the instances whose stalks are enlarging, while the target data set consists of the instances whose stalks are tapering.

For 20newsgroups data (Task2, Task3 and Task4) contains seven top categories and 20 subcategories. The task is defined as the top-category-classification problems. We split the data based on the subcategories. For example, in Task2, we learn to classify two classes, sci and talk. The target data set consists of two subcategories sci.space and talk.religion,misc, while the source data set consists of all other subcategories data under the top categories sci and talk, i.e. sci.crypt, sci.electronic, sci.med, talk.politics.guns, talk.politics.mideast and talk.politics.misc. Task3 and Task4 are similar to Task2.

Table 1. Data sets description

Task	Data set	Size	
		$D_l \bigcup D_u$	D_s
1	edibles vs poisonous	4608	3156
2	sci.space vs talk.religion.misc	4880	2315
3	rec.sport.hockey vs sci.space	5089	2537
4	rec.sport.hockey vs talk.religion.misc	4883	2320

[1] http://archive.ics.uci.edu/ml/datasets/Mushroom

[2] http://qwone.com/~jason/20Newsgroups

4.2 Comparison Methods

We compare our algorithm with four algorithms: SVM trained on only the target training data D_l, SVMt and TrAdaBoost trained on both target and source training data D_s, and PTSVM trained on both target labeled data D_l and unlabeled data D_u.

SVM^{light} [25] is used as the basic learner in our experiments. When training SVM, the training data set only consists of the target domain labeled data which is not enough to train a good classifier. Except for target domain labeled data, SVMt also uses source domain labeled date D_s as additional training set. Besides, SVMt is the same as SVM. So the result of SVMt reflects the effect of source data. The training set used in TrAdaBoost is the same as SVMt, but it adjusts the weights of all training instances through interations. So the benefit of TrAdaBoost is brought by transfer learning compare to SVMt. PTSVM selects the instances from the unlabeled target data set D_u, and adds them to D_l.

Our TPTSVM algorithm is trained on the three data sets: D_l, D_s and D_u, trying to improve the transfer learning with semi-supervised learning.

We also tried to experiment with two other methods, Frustratingly Easy Semi-Supervised Domain Adaptation [26] and TrAdaBoost employing TSVM, which also training with unlabeled data, but the results was not satisfied on our data sets.

4.3 Experiment Results

The parameters of SVM are set to the default values of SVM^{light}. Iteration time N of both TrAdaBoost and TPTSVM is 100. The growth size p at each iteration in TPTSVM is set to 20. All the results below are the average of 10 repeats by random.

The results on the mushroom data are showed in Figure 1. To investigate the effect of the target training size to each algorithm, we use 50, 100 and 150 target instances as training set respectively (positive and negative instances are half and half). From the results we can see, when the training data is not sufficient (50 instances), the performance of SVM is poor. The knowledge from the source data may help the SVMt learner. While when training instance number is 150, it is enough to train a good SVM classifier. The additional training set of source domain takes negative effect to the SVMt learner. Therefore, the source training data contain not only good knowledge, but also noisy data. TrAdaBoost could always perform better than SVMt and

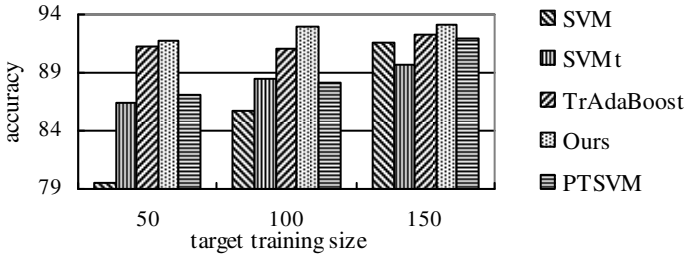

Fig. 1. Accuracy on mushroom data

SVM, because it could reduce the effect of the noisy source data, while transfer useful knowledge to the target domain. PTSVM also improves the accuracy of SVM by using the target unlabeled data to help learning. Our algorithm gives the best performance steadily through the novel combination of transfer and transductive learning.

As for Task 2, figure 2 compares the test classification accuracy of the algorithms when training on the different size of the target training set. It can be seen that the results with different training size show similar trend as Task1 on mushroom data. When the number of training set is 50, TrAdaBoost performs worse than SVMt. But we found that the accuracy on target domain training set is 100%. This may be cause by the over-fitting of AdaBoost. Our algorithm also uses the boosting of AdaBoost, but our algorithm could effectively overcome the over-fitting of AdaBoost by semi-supervised learning. Figure 3 shows the accuracy curves of TrAdaBoost and our algorithm with different numbers of iterations. The size of the target training set is 400. From the curves, we can see how semi-supervised learning helps to improve transfer learning in normal situation.

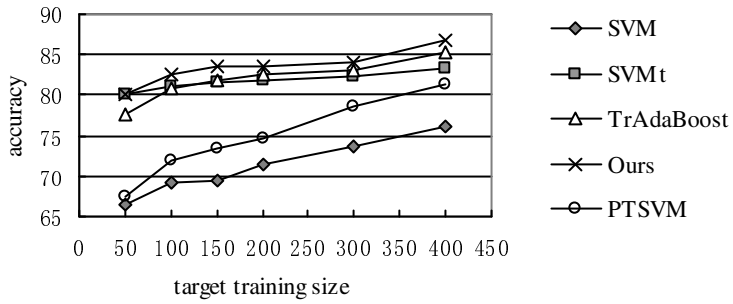

Fig. 2. Accuracy on Task2

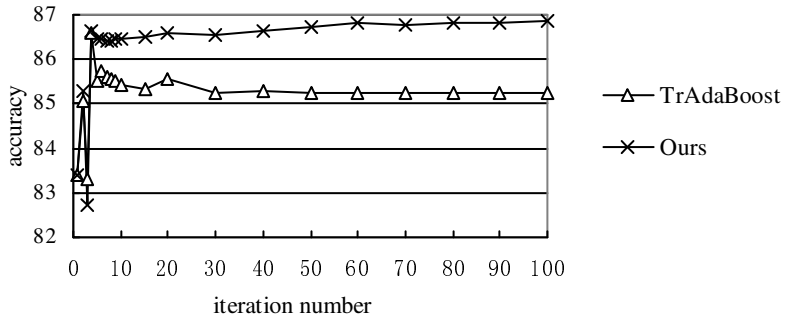

Fig. 3. Learning curves on Task2

Table 2 shows comparison results on test classification tasks (Task2, Task3 and Task4). We do the same experiment on them. The number of training instances is 150. Seen from the table, our algorithm outperforms all four algorithms over all test classification tasks.

Table 2. Comparison results on test classification tasks

Algorithm	Task 2	Task 3	Task 4
SVM	69.42	56.17	68.83
SVMt	81.58	62.20	79.40
TrAdaBoost	81.84	63.25	80.52
PTSVM	73.40	57.35	66.36
Ours	83.47	63.86	81.25

5 Conclusion

TPTSVM is a novel approach to knowledge transfer and domain adaptation by combining transfer learning and semi-supervised learning. The proposed TPTSVM algorithm transfers the source knowledge to the target domain, and further adapts the classification function to the target domain through some unlabeled target data. The theoretical analysis demonstrates its improved performance against transfer and semi-supervised algorithm. And the experiments confirm its better transferability especially for little target training data. Extending our algorithm to multiple source domains are left to the future work.

Acknowledgments. This work is supported by the National Fundamental Research Program of China (61272375, 61173100 and 61173101).

References

1. Yin, X., Han, J., Yang, J., et al.: Efficient classification across multiple database relations: A crossmine approach. IEEE Transactions on Knowledge and Data Engineering 18(6), 770–783 (2006)
2. Kuncheva, L.I., Rodriguez, J.J.: Classifier ensembles with a random linear oracle. IEEE Transactions on Knowledge and Data Engineering 19(4), 500–508 (2007)
3. Baralis, E., Chiusano, S., Garza, P.: A lazy approach to associative classification. IEEE Transactions on Knowledge and Data Engineering 20(2), 156–171 (2008)
4. Fung, G.P.C., Yu, J.X., Lu, H., et al.: Text classification without negative examples revisit. IEEE Transactions on Knowledge and Data Engineering 18(1), 6–20 (2006)
5. Al-Mubaid, H., Umair, S.A.: A new text categorization technique using distributional clustering and learning logic. IEEE Transactions on Knowledge and Data Engineering 18(9), 1156–1165 (2006)
6. Sarinnapakorn, K., Kubat, M.: Combining subclassifiers in text categorization: A dst-based solution and a case study. IEEE Transactions on Knowledge and Data Engineering 19(12), 1638–1651 (2007)
7. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)

8. Arnold, A., Nallapati, R., Cohen, W.W.: A comparative study of methods for transductive transfer learning. In: Seventh IEEE International Conference on ICDM Workshops 2007, pp. 77–82 (2007)
9. Thrun, S., Mitchell, T.M.: Learning one more thing. R. Carnegie-Mellon Univ. Pittsburgh Pa Dept. of Computer Science (1994)
10. Schmidhuber, J.: On learning how to learn learning strategies (1995)
11. Caruana, R.: Multitask learning. Springer, US (1998)
12. Blitzer, J., McDonald, R., Pereira, F.: Domain adaptation with structural correspondence learning. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 120–128. Association for Computational Linguistics (2006)
13. Daumé III, H., Marcu, D.: Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research 26(1), 101–126 (2006)
14. Pan, S.J., Zheng, V.W., Yang, Q., et al.: Transfer learning for wifi-based indoor localization. In: Association for the Advancement of Artificial Intelligence (AAAI) Workshop (2008)
15. Rosenstein, M.T., Marx, Z., Kaelbling, L.P., et al.: To transfer or not to transfer. In: NIPS 2005 Workshop on Transfer Learning, p. 898 (2005)
16. Dai, W., Yang, Q., Xue, G.R., et al.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning, pp. 193–200. ACM (2007)
17. Eaton, E., des Jardins, M.: Selective transfer between learning tasks using task-based boosting. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
18. Luo, C., Ji, Y., Dai, X., et al.: Active learning with transfer learning. In: Proceedings of ACL 2012 Student Research Workshop, pp. 13–18. Association for Computational Linguistics (2012)
19. Shao, M., Castillo, C., Gu, Z., et al.: Low-Rank Transfer Subspace Learning. In: Twelfth IEEE International Conference on ICDM Workshops 2012, pp. 1104–1109 (2012)
20. Negahban, S.N., Rubinstein, B.I.P., Gemmell, J.G.: Scaling multiple-source entity resolution using statistically efficient transfer learning. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 2224–2228. ACM (2012)
21. Ju, S., Li, S., Su, Y., et al.: Dual word and document seed selection for semi-supervised sentiment classification. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 2295–2298. ACM (2012)
22. Joachims, T.: Transductive inference for text classification using support vector machines. In: Machine Learning-International Workshop Then Conference, pp. 200–209. Morgan Kaufmann Publishers, Inc. (2009)
23. Chen, Y., Wang, G., Dong, S.: Learning with progressive transductive support vector machine. Pattern Recognition Letters 24(12), 1845–1855 (2003)
24. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
25. Joachims, T.: Learning to classify text using support vector machines: Methods, theory and algorithms. Kluwer Academic Publishers (2002)
26. Daumé III, H., Kumar, A., Saha, A.: Frustratingly easy semi-supervised domain adaptation. In: Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing, pp. 53–59. Association for Computational Linguistics (2010)