Instance Weighting for Domain Adaptation in NLP

Jing Jiang and **ChengXiang Zhai**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{jiang4,czhai}@cs.uiuc.edu

Abstract

Domain adaptation is an important problem in natural language processing (NLP) due to the lack of labeled data in novel domains. In this paper, we study the domain adaptation problem from the instance weighting perspective. We formally analyze and characterize the domain adaptation problem from a distributional view, and show that there are two distinct needs for adaptation, corresponding to the different distributions of instances and classification functions in the source and the target domains. We then propose a general instance weighting framework for domain adaptation. Our empirical results on three NLP tasks show that incorporating and exploiting more information from the target domain through instance weighting is effective.

1 Introduction

Many natural language processing (NLP) problems such as part-of-speech (POS) tagging, named entity (NE) recognition, relation extraction, and semantic role labeling, are currently solved by supervised learning from manually labeled data. A bottleneck problem with this supervised learning approach is the lack of annotated data. As a special case, we often face the situation where we have a sufficient amount of labeled data in one domain (call the *source* domain), but have little or no labeled data in another related domain (called the *target* domain) which we are interested in. We thus face the *domain adaptation* problem.

The domain adaptation problem is commonly encountered in NLP. For example, in POS tagging, the source domain may be tagged WSJ articles, and the target domain may be scientific literature that contains scientific terminology. In NE recognition, the source domain may be annotated news articles, and the target domain may be personal blogs. Another example is personalized spam filtering, where we may have many labeled spam and ham emails from publicly available sources, but we need to adapt the learned spam filter to an individual user's inbox because the user has her own, and presumably very different, distribution of emails and notion of spams.

Despite the importance of domain adaptation in NLP, currently there are no standard methods for solving this problem. An immediate possible solution is semi-supervised learning, where we simply treat the target instances as unlabeled data but do not distinguish the two domains. However, given that the source data and the target data are from different distributions, we should expect to do better by exploiting the domain difference. Recently there have been some studies addressing domain adaptation from different perspectives (Roark and Bacchiani, 2003; Chelba and Acero, 2004; Florian et al., 2004; Daumé III and Marcu, 2006; Blitzer et al., 2006). However, there have not been many studies that focus on the difference between the instance distributions in the two domains. A detailed discussion on related work is given in Section 5.

In this paper, we study the domain adaptation problem from the instance weighting perspective. In general, the domain adaptation problem arises when the source instances and the target instances are from two different, but related distributions. We formally analyze and characterize the domain adaptation problem from this distributional view. Such an analysis reveals that there are two distinct needs for adaptation, corresponding to the different distributions of instances and the different classification functions in the source and the target domains. Based on this analysis, we propose a general instance weighting method for domain adaptation, which can be regarded as a generalization of an existing approach to semi-supervised learning. The proposed method implements several adaptation heuristics with a unified objective function: (1) removing misleading training instances in the source domain; (2) assigning more weights to labeled target instances than labeled source instances; (3) augmenting training instances with target instances with predicted labels. We evaluated the proposed method with three adaptation problems in NLP, including POS tagging, NE type classification, and spam filtering. The results show that regular semi-supervised and supervised learning methods do not perform as well as our new method, which explicitly captures domain difference. Our results also show that incorporating and exploiting more information from the target domain is much more useful for improving performance than excluding misleading training examples from the source domain.

The rest of the paper is organized as follows. In Section 2, we formally analyze the domain adaptation problem and distinguish two types of adaptation. In Section 3, we then propose a general instance weighting framework for domain adaptation. In Section 4, we present the experiment results. Finally, we compare our framework with related work in Section 5 before we conclude in Section 6.

2 Domain Adaptation

In this section, we define and analyze domain adaptation from a theoretical point of view. We show that the need for domain adaptation arises from two factors, and the solutions are different for each factor. We restrict our attention to those NLP tasks that can be cast into multiclass classification problems, and we only consider discriminative models for classification. Since both are common practice in NLP, our analysis is applicable to many NLP tasks.

Let $\mathcal{X}$ be a feature space we choose to represent the observed instances, and let $\mathcal{Y}$ be the set of class labels. In the standard supervised learning setting, we are given a set of labeled instances $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and (x_i, y_i) are drawn from an unknown joint distribution $p(x, y)$. Our goal is to recover this unknown distribution so that we can predict unlabeled instances drawn from the same distribution. In discriminative models, we are only concerned with $p(y|x)$. Following the maximum likelihood estimation framework, we start with a parameterized model family $p(y|x;\theta)$, and then find the best model parameter θ^* that maximizes the expected log likelihood of the data:

$$\theta^* = \arg\max_{\theta} \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x;\theta) dx.$$

Since we do not know the distribution $p(x, y)$, we maximize the empirical log likelihood instead:

$$\begin{aligned} \theta^* &\approx \arg\max_{\theta} \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} \tilde{p}(x, y) \log p(y|x;\theta) dx \\ &= \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i;\theta). \end{aligned}$$

Note that since we use the empirical distribution $\tilde{p}(x, y)$ to approximate $p(x, y)$, the estimated θ^* is dependent on $\tilde{p}(x, y)$. In general, as long as we have sufficient labeled data, this approximation is fine because the unlabeled instances we want to classify are from the same $p(x, y)$.

2.1 Two Factors for Domain Adaptation

Let us now turn to the case of domain adaptation where the unlabeled instances we want to classify are from a different distribution than the labeled instances. Let $p_s(x, y)$ and $p_t(x, y)$ be the true underlying distributions for the source and the target domains, respectively. Our general idea is to use $p_s(x, y)$ to approximate $p_t(x, y)$ so that we can exploit the labeled examples in the source domain.

If we factor $p(x, y)$ into $p(x, y) = p(y|x)p(x)$, we can see that $p_t(x, y)$ can deviate from $p_s(x, y)$ in two different ways, corresponding to two different kinds of domain adaptation:

Case 1 (Labeling Adaptation): $p_t(y|x)$ deviates from $p_s(y|x)$ to a certain extent. In this case, it is clear that our estimation of $p_s(y|x)$ from the labeled source domain instances will not be a good estimation of $p_t(y|x)$, and therefore domain adaptation is needed. We refer to this kind of adaptation as *function/labeling adaptation*.

Case 2 (Instance Adaptation): $p_t(y|x)$ is mostly similar to $p_s(y|x)$, but $p_t(x)$ deviates from $p_s(x)$. In this case, it may appear that our estimated $p_s(y|x)$ can still be used in the target domain. However, as we have pointed out, the estimation of $p_s(y|x)$ depends on the empirical distribution $\tilde{p}_s(x, y)$, which deviates from $p_t(x, y)$ due to the deviation of $p_s(x)$ from $p_t(x)$. In general, the estimation of $p_s(y|x)$ would be more influenced by the instances with high $\tilde{p}_s(x, y)$ (i.e., high $\tilde{p}_s(x)$). If $p_t(x)$ is very different from $p_s(x)$, then we should expect $p_t(x, y)$ to be very different from $p_s(x, y)$, and therefore different from $\tilde{p}_s(x, y)$. We thus cannot expect the estimated $p_s(y|x)$ to work well on the regions where $p_t(x, y)$ is high, but $p_s(x, y)$ is low. Therefore, in this case, we still need domain adaptation, which we refer to as *instance adaptation*.

Because the need for domain adaptation arises from two different factors, we need different solutions for each factor.

2.2 Solutions for Labeling Adaptation

If $p_t(y|x)$ deviates from $p_s(y|x)$ to some extent, we have one of the following choices:

Change of representation:

It may be the case that if we change the representation of the instances, i.e., if we choose a feature space $\mathcal{X}'$ different from $\mathcal{X}$, we can bridge the gap between the two distributions $p_s(y|x)$ and $p_t(y|x)$. For example, consider domain adaptive NE recognition where the source domain contains clean newswire data, while the target domain contains broadcast news data that has been transcribed by automatic speech recognition and lacks capitalization. Suppose we use a naive NE tagger that only looks at the word itself. If we consider capitalization, then the instance *Bush* is represented differently from the instance *bush*. In the source domain, $p_s(y = \text{Person}|x = \textit{Bush})$ is high while $p_s(y = \text{Person}|x = \textit{bush})$ is low, but in the target domain, $p_t(y = \text{Person}|x = \textit{bush})$ is high. If we ignore the capitalization information, then in both domains $p(y = \text{Person}|x = \textit{bush})$ will be high provided that the source domain contains much fewer instances of *bush* than *Bush*.

Adaptation through prior:

When we use a parameterized model $p(y|x; \theta)$ to approximate $p(y|x)$ and estimate θ based on the source domain data, we can place some prior on the model parameter θ so that the estimated distribution $p(y|x; \hat{\theta})$ will be closer to $p_t(y|x)$. Consider again the NE tagging example. If we use capitalization as a feature, in the source domain where capitalization information is available, this feature will be given a large weight in the learned model because it is very useful. If we place a prior on the weight for this feature so that a large weight will be penalized, then we can prevent the learned model from relying too much on this domain specific feature.

Instance pruning:

If we know the instances x for which $p_t(y|x)$ is different from $p_s(y|x)$, we can actively remove these instances from the training data because they are “misleading”.

For all the three solutions given above, we need either some prior knowledge about the target domain, or some labeled target domain instances; from only the unlabeled target domain instances, we would not know where and why $p_t(y|x)$ differs from $p_s(y|x)$.

2.3 Solutions for Instance Adaptation

In the case where $p_t(y|x)$ is similar to $p_s(y|x)$, but $p_t(x)$ deviates from $p_s(x)$, we may use the (unlabeled) target domain instances to bias the estimate of $p_s(x)$ toward a better approximation of $p_t(x)$, and thus achieve domain adaptation. We explain the idea below.

Our goal is to obtain a good estimate of θ_t^* that is optimized according to the *target* domain distribution $p_t(x, y)$. The exact objective function is thus

$$\begin{aligned}\theta_t^* &= \arg\max_{\theta} \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} p_t(x, y) \log p(y|x; \theta) dx \\ &= \arg\max_{\theta} \int_{\mathcal{X}} p_t(x) \sum_{y \in \mathcal{Y}} p_t(y|x) \log p(y|x; \theta) dx.\end{aligned}$$

Our idea of domain adaptation is to exploit the labeled instances in the *source* domain to help obtain

θ_t^*.

Let $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s}$ denote the set of labeled instances we have from the source domain. Assume that we have a (small) set of labeled and a (large) set of unlabeled instances from the target domain, denoted by $\mathcal{D}_{t,l} = \{(x_j^{t,l}, y_j^{t,l})\}_{j=1}^{N_{t,l}}$ and $\mathcal{D}_{t,u} = \{x_k^{t,u}\}_{k=1}^{N_{t,u}}$, respectively. We now show three ways to approximate the objective function above, corresponding to using three different sets of instances to approximate the instance space $\mathcal{X}$.

Using $\mathcal{D}_s$:

Using $p_s(y|x)$ to approximate $p_t(y|x)$, we obtain

$$\begin{aligned}\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \frac{p_t(x)}{p_s(x)} p_s(x) \sum_{y\in\mathcal{Y}} p_s(y|x) \log p(y|x;\theta) dx \\ &\approx \arg\max_\theta \int_{\mathcal{X}} \frac{p_t(x)}{p_s(x)} \tilde{p}_s(x) \sum_{y\in\mathcal{Y}} \tilde{p}_s(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{p_t(x_i^s)}{p_s(x_i^s)} \log p(y_i^s|x_i^s;\theta).\end{aligned}$$

Here we use only the labeled instances in $\mathcal{D}_s$ but we adjust the weight of each instance by $\frac{p_t(x)}{p_s(x)}$. The major difficulty is how to accurately estimate $\frac{p_t(x)}{p_s(x)}$.

Using $\mathcal{D}_{t,l}$:

$$\begin{aligned}\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \tilde{p}_{t,l}(x) \sum_{y\in\mathcal{Y}} \tilde{p}_{t,l}(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y_j^{t,l}|x_j^{t,l};\theta).\end{aligned}$$

Note that this is the standard supervised learning method using only the small amount of labeled target instances. The major weakness of this approximation is that when $N_{t,l}$ is very small, the estimation is not accurate.

Using $\mathcal{D}_{t,u}$:

$$\begin{aligned}\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \tilde{p}_{t,u}(x) \sum_{y\in\mathcal{Y}} p_t(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_\theta \frac{1}{N_{t,u}} \sum_{k=1}^{N_{t,u}} \sum_{y\in\mathcal{Y}} p_t(y|x_k^{t,u}) \log p(y|x_k^{t,u};\theta).\end{aligned}$$

The challenge here is that $p_t(y|x_k^{t,u};\theta)$ is unknown to us, and thus we need to estimate it. One possibility is to approximate it with a model $\hat{\theta}$ learned from $\mathcal{D}_s$ and $\mathcal{D}_{t,l}$. For example, we can set $p_t(y|x,\theta) = p(y|x;\hat{\theta})$. Alternatively, we can also set $p_t(y|x,\theta)$ to 1 if $y = \arg\max_{y'} p(y'|x;\hat{\theta})$ and 0 otherwise.

3 A Framework of Instance Weighting for Domain Adaptation

The theoretical analysis we give in Section 2 suggests that one way to solve the domain adaptation problem is through instance weighting. We propose a framework that incorporates instance pruning in Section 2.2 and the three approximations in Section 2.3. Before we show the formal framework, we first introduce some weighting parameters and explain the intuitions behind these parameters.

First, for each $(x_i^s, y_i^s) \in \mathcal{D}_s$, we introduce a parameter α_i to indicate how likely $p_t(y_i^s|x_i^s)$ is close to $p_s(y_i^s|x_i^s)$. Large α_i means the two probabilities are close, and therefore we can trust the labeled instance (x_i^s, y_i^s) for the purpose of learning a classifier for the target domain. Small α_i means these two probabilities are very different, and therefore we should probably discard the instance (x_i^s, y_i^s) in the learning process.

Second, again for each $(x_i^s, y_i^s) \in \mathcal{D}_s$, we introduce another parameter β_i that ideally is equal to $\frac{p_t(x_i^s)}{p_s(x_i^s)}$. From the approximation in Section 2.3 that uses only $\mathcal{D}_s$, it is clear that the accuracy of such an approximation is mostly affected by the accuracy of our estimate of α_i's and β_i's.

Next, for each $x_i^{t,u} \in \mathcal{D}_{t,u}$, and for each possible label $y \in \mathcal{Y}$, we introduce a parameter $\gamma_i(y)$ that indicates how likely we would like to assign y as a tentative label to $x_i^{t,u}$ and include $(x_i^{t,u}, y)$ as a training example.

Finally, we introduce three global parameters λ_s, $\lambda_{t,l}$ and $\lambda_{t,u}$ that are not instance-specific but are associated with $\mathcal{D}_s$, $\mathcal{D}_{t,l}$ and $\mathcal{D}_{t,u}$, respectively. These three parameters allow us to control the contribution of each of the three approximation methods in Section 2.3 when we linearly combine them together.

We now formally define our instance weighting framework. Given $\mathcal{D}_s$, $\mathcal{D}_{t,l}$ and $\mathcal{D}_{t,u}$, to learn a classifier for the target domain, we find a parameter $\hat{\theta}$ that optimizes the following objective function, in which we use all the available instances to approximate the instance space $\mathcal{X}$:

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \Bigg[\lambda_s \cdot \frac{1}{C_s}\sum_{i=1}^{N_s} \alpha_i\beta_i \log p(y_i^s|x_i^s;\theta) \\ &\quad +\lambda_{t,l}\cdot\frac{1}{C_{t,l}}\sum_{j=1}^{N_{t,l}} \log p(y_j^{t,l}|x_j^{t,l};\theta) \\ &\quad +\lambda_{t,u}\cdot\frac{1}{C_{t,u}}\sum_{k=1}^{N_{t,u}}\sum_{y\in\mathcal{Y}} \gamma_k(y)\log p(y|x_k^{t,u};\theta) \\ &\quad +\log p(\theta)\Bigg],\end{aligned}$$

where $C_s = \sum_{i=1}^{N_s} \alpha_i\beta_i$, $C_{t,l} = N_{t,l}$, $C_{t,u} = \sum_{k=1}^{N_{t,u}}\sum_{y\in\mathcal{Y}}\gamma_k(y)$, and $\lambda_s + \lambda_{t,l} + \lambda_{t,u} = 1$. The last term, $\log p(\theta)$, is the log of a Gaussian prior distribution of θ, commonly used to regularize the complexity of the model.

In general, we do not know the optimal values of these parameters for the target domain. Nevertheless, the intuitions behind these parameters serve as guidelines for us to design heuristics to set these parameters. In the rest of this section, we introduce several heuristics that we used in our experiments to set these parameters.

3.1 Setting α

Following the intuition that if $p_t(y|x)$ differs much from $p_s(y|x)$, then (x, y) should be discarded from the training set, we use the following heuristic to set α^s. First, with standard supervised learning, we train a model $\hat{\theta}_{t,l}$ from $\mathcal{D}_{t,l}$. We consider $p(y|x;\hat{\theta}_{t,l})$ to be a crude approximation of $p_t(y|x)$. Then, we classify $\{x_i^s\}_{i=1}^{N_s}$ using $\hat{\theta}_{t,l}$. The top k instances that are incorrectly predicted by $\hat{\theta}_{t,l}$ (ranked by their prediction confidence) are discarded. In another word, α_i^s of the top k instances for which $y_i^s \neq \arg\max_y p(y|x_i^s;\hat{\theta}_{t,l})$ are set to 0, and α_i of all the other source instances are set to 1.

3.2 Setting β

Accurately setting β involves accurately estimating $p_s(x)$ and $p_t(x)$ from the empirical distributions. For many NLP classification tasks, we do not have a good parametric model for $p(x)$. We could resort to some non-parametric density estimation methods. However, we have not explored this direction. In our experiments, we set β to 1 for all source instances.

3.3 Setting γ

Setting γ is closely related to some semi-supervised learning methods. One option is to set $\gamma_k(y) = p(y|x_k^{t,u};\theta)$. In this case, γ is no longer a constant but is a function of θ. This way of setting γ corresponds to the entropy minimization semi-supervised learning method (Grandvalet and Bengio, 2005). Another way to set γ corresponds to bootstrapping semi-supervised learning. First, let $\hat{\theta}^{(n)}$ be a model learned from the previous round of training. We then select the top k instances from $\mathcal{D}_{t,u}$ that have the highest prediction confidence. For these instances, we set $\gamma_k(y) = 1$ for $y = \arg\max_{y'} p(y'|x_k^{t,u};\hat{\theta}^{(n)})$, and $\gamma_k(y) = 0$ for all other y. In another word, we select the top k confidently predicted instances, and include these instances together with their predicted labels in the training set. All other instances in $\mathcal{D}_{t,u}$ are not considered. In our experiments, we only considered this bootstrapping way of setting γ.

3.4 Setting λ

λ_s, $\lambda_{t,l}$ and $\lambda_{t,u}$ control the balance among the three sets of instances. Using standard supervised learning, λ_s and $\lambda_{t,l}$ are set proportionally to C_s and $C_{t,l}$, that is, each instance is weighted the same whether it is in $\mathcal{D}_s$ or in $\mathcal{D}_{t,l}$, and $\lambda_{t,u}$ is set to 0. Similarly, using standard bootstrapping, $\lambda_{t,u}$ is set proportionally to $C_{t,u}$, that is, each target instance added to the training set is also weighted the same as a source instance. In neither case are the target instances emphasized more than source instances. However, for domain adaptation, we want to focus more on the target domain instances. So intuitively, we want to make $\lambda_{t,l}$ and $\lambda_{t,u}$ somehow larger relative to λ_s. As we will show in Section 4, this is indeed beneficial.

In general, the framework provides great flexibility for implementing different adaptation strategies through these instance weighting parameters.

4 Experiments

4.1 Tasks and Data Sets

We chose three different NLP tasks to evaluate our instance weighting method for domain adaptation. The first task is POS tagging, for which we used 6166 WSJ sentences from Sections 00 and 01 of Penn Treebank as the source domain data, and 2730 PubMed sentences from the Oncology section of the

PennBioIE corpus as the target domain data.[1] The second task is entity type classification. The setup is very similar to Daumé III and Marcu (2006). We assume that the entity boundaries have been correctly identified, and we want to classify the types of the entities. We used ACE 2005 training data for this task.[2] For the source domain, we used the newswire collection, which contains 11256 examples, and for the target domains, we used the weblog (WL) collection (5164 examples) and the conversational telephone speech (CTS) collection (4868 examples). The third task is personalized spam filtering. We used the ECML/PKDD 2006 discovery challenge task A data set.[3] The source domain contains 4000 spam and ham emails from publicly available sources, and the target domains are three individual users' inboxes, each containing 2500 emails.

For each task, we consider two experiment settings. In the first setting, we assume there are a small number of labeled target instances available. For POS tagging, we used an additional 300 Oncology sentences as labeled target instances. For NE typing, we used 500 labeled target instances and 2000 unlabeled target instances for each target domain. For spam filtering, we used 200 labeled target instances and 1800 unlabeled target instances. In the second setting, we assume there is no labeled target instance. We thus used all available target instances for testing in all three tasks.

We used logistic regression as our model of $p(y|x;\theta)$ because it is a robust learning algorithm and widely used. We used classification accuracy as the primary performance measure for comparing different methods.

We now describe three sets of experiments, corresponding to three heuristic ways of setting α, $\lambda_{t,l}$ and $\lambda_{t,u}$.

4.2 Removing "Misleading" Source Domain Instances

In the first set of experiments, we let the small number of labeled target instances guide us to gradually remove potentially "misleading" labeled instances from the source domain. We follow the heuristic we described in Section 3.1, which sets the α for the top k misclassified source instances to 0, and the α for all the other source instances to 1. We also set $\lambda_{t,l}$ and $\lambda_{t,l}$ to 0 in order to focus only on the effect of removing "misleading" instances. We compare with a baseline method which uses all source instances with equal weight. The results are shown in Table 1.

From the table, we can see that in most experiments, removing these predicted "misleading" examples improved the performance over the baseline. In some experiments (Oncology, CTS, u00, u01), the largest improvement was achieved when all misclassified source instances were removed. In the case of weblog NE type classification, however, removing the source instances hurt the performance. A possible reason for this is that the set of labeled target instances we use is a biased sample from the target domain, and therefore the model trained on these instances is not always a good predictor of "misleading" source instances.

4.3 Adding Labeled Target Domain Instances with Higher Weights

The second set of experiments is to add the labeled target domain instances into the training set. This corresponds to setting $\lambda_{t,l}$ to some non-zero value, but still keeping $\lambda_{t,u}$ as 0. If we ignore the domain difference, then each labeled target instance is weighted the same as a labeled source instance ($\frac{\lambda_{u,l}}{\lambda_s} = \frac{C_{u,l}}{C_s}$), which is what happens in regular supervised learning. However, based on our theoretical analysis, we can expect the labeled target instances to be more representative of the target domain than the source instances. We can therefore assign higher weights for the target instances, by adjusting the ratio between $\lambda_{t,l}$ and λ_s. In our experiments, we set $\frac{\lambda_{t,l}}{\lambda_s} = a\frac{C_{t,l}}{C_s}$, where a ranges from 2 to 20. The results are shown in Table 2.

As shown from the table, adding some labeled target instances can greatly improve the performance for all tasks. And in almost all cases, weighting the target instances more than the source instances performed better than weighting them equally.

We also tested another setting where we first removed the "misleading" source examples as we showed in Section 4.2, and then added the labeled target instances. The results are shown in the last row of Table 2. However, although both removing

[1] http://bioie.ldc.upenn.edu/publications/latest_release/data/

[2] http://www.nist.gov/speech/tests/ace/ace05/

[3] http://www.ecmlpkdd2006.org/challenge.html

POS		NE Type				Spam			
k	**Oncology**	k	**CTS**	k	**WL**	k	**u00**	**u01**	**u02**
0	0.8630	0	0.7815	0	0.7045	0	0.6306	0.6950	0.7644
4000	0.8675	800	0.8245	600	0.7070	150	0.6417	0.7078	0.7950
8000	0.8709	1600	0.8640	1200	0.6975	300	0.6611	0.7228	0.8222
12000	0.8713	2400	0.8825	1800	0.6830	450	0.7106	0.7806	0.8239
16000	0.8714	3000	0.8825	2400	0.6795	600	0.7911	0.8322	0.8328
all	0.8720	all	0.8830	all	0.6600	all	0.8106	0.8517	0.8067

Table 1: Accuracy on the target domain after removing "misleading" source domain instances.

POS		NE Type			Spam			
method	**Oncology**	**method**	**CTS**	**WL**	**method**	**u00**	**u01**	**u02**
$\mathcal{D}_s$ only	0.8630	$\mathcal{D}_s$ only	0.7815	0.7045	$\mathcal{D}_s$ only	0.6306	0.6950	0.7644
$\mathcal{D}_s + \mathcal{D}_{t,l}$	0.9349	$\mathcal{D}_s + \mathcal{D}_{t,l}$	0.9340	0.7735	$\mathcal{D}_s + \mathcal{D}_{t,l}$	0.9572	0.9572	0.9461
$\mathcal{D}_s + 5\mathcal{D}_{t,l}$	0.9411	$\mathcal{D}_s + 2\mathcal{D}_{t,l}$	0.9355	0.7810	$\mathcal{D}_s + 2\mathcal{D}_{t,l}$	0.9606	0.9600	0.9533
$\mathcal{D}_s + 10\mathcal{D}_{t,l}$	0.9429	$\mathcal{D}_s + 5\mathcal{D}_{t,l}$	0.9360	0.7820	$\mathcal{D}_s + 5\mathcal{D}_{t,l}$	0.9628	09611	0.9601
$\mathcal{D}_s + 20\mathcal{D}_{t,l}$	0.9443	$\mathcal{D}_s + 10\mathcal{D}_{t,l}$	0.9355	0.7840	$\mathcal{D}_s + 10\mathcal{D}_{t,l}$	0.9639	0.9628	0.9633
$\mathcal{D}'_s + 20\mathcal{D}_{t,l}$	0.9422	$\mathcal{D}'_s + 10\mathcal{D}_{t,l}$	0.8950	0.6670	$\mathcal{D}'_s + 10\mathcal{D}_{t,l}$	0.9717	0.9478	0.9494

Table 2: Accuracy on the unlabeled target instances after adding the labeled target instances.

"misleading" source instances and adding labeled target instances work well individually, when combined, the performance in most cases is not as good as when no source instances are removed. We hypothesize that this is because after we added some labeled target instances with large weights, we already gained a good balance between the source data and the target data. Further removing source instances would push the emphasis more on the set of labeled target instances, which is only a biased sample of the whole target domain.

The POS data set and the CTS data set have previously been used for testing other adaptation methods (Daumé III and Marcu, 2006; Blitzer et al., 2006), though the setup there is different from ours. Our performance using instance weighting is comparable to their best performance (slightly worse for POS and better for CTS).

4.4 Bootstrapping with Higher Weights

In the third set of experiments, we assume that we do not have any labeled target instances. We tried two bootstrapping methods. The first is a *standard* bootstrapping method, in which we gradually added the most confidently predicted unlabeled target instances with their predicted labels to the training set. Since we believe that the target instances should in general be given more weight because they better represent the target domain than the source instances, in the second method, we gave the added target instances more weight in the objective function. In particular, we set $\lambda_{t,u} = \lambda_s$ such that the total contribution of the added target instances is equal to that of all the labeled source instances. We call this second method the *balanced* bootstrapping method. Table 3 shows the results.

As we can see, while bootstrapping can generally improve the performance over the baseline where no unlabeled data is used, the balanced bootstrapping method performed slightly better than the standard bootstrapping method. This again shows that weighting the target instances more is a right direction to go for domain adaptation.

5 Related Work

There have been several studies in NLP that address domain adaptation, and most of them need labeled data from both the source domain and the target domain. Here we highlight a few representative ones.

For generative syntactic parsing, Roark and Bacchiani (2003) have used the source domain data to construct a Dirichlet prior for MAP estimation of the PCFG for the target domain. Chelba and Acero (2004) use the parameters of the maximum entropy model learned from the source domain as the means of a Gaussian prior when training a new model on the target data. Florian et al. (2004) first train a NE tagger on the source domain, and then use the tagger's predictions as features for training and testing on the target domain.

	POS	NE Type		Spam		
method	**Oncology**	**CTS**	**WL**	**u00**	**u01**	**u02**
supervised	0.8630	0.7781	0.7351	0.6476	0.6976	0.8068
standard bootstrap	0.8728	0.8917	0.7498	0.8720	0.9212	0.9760
balanced bootstrap	0.8750	0.8923	0.7523	0.8816	0.9256	0.9772

Table 3: Accuracy on the target domain without using labeled target instances. In balanced bootstrapping, more weights are put on the target instances in the objective function than in standard bootstrapping.

The only work attempting to model the different distributions in the source and target domains that we are aware of is Daumé III and Marcu (2006). They assume a "true source domain" distribution, a "true target domain" distribution, and a "general domain" distribution. The source (target) domain data is generated from a mixture of the "truly source (target) domain" distribution and the "general domain" distribution. In contrast, we do not assume such a mixture model.

None of the above methods would work if there were no labeled target instances. Indeed, all the above methods do not make use of the unlabeled instances in the target domain. In contrast, our instance weighting framework allows unlabeled target instances to contribute to the model estimation.

Blitzer et al. (2006) propose a domain adaptation method that uses the unlabeled target instances to infer a good feature representation, which can be regarded as weighting the features. In contrast, we weight the instances. The idea of using $\frac{p_t(x)}{p_s(x)}$ to weight instances has been studied in statistics (Shimodaira, 2000), but has not been applied to NLP tasks.

6 Conclusions and Future Work

In this paper, we formally analyze the domain adaptation problem and propose a general instance weighting framework for domain adaptation. The framework is flexible to support many different strategies for adaptation. In particular, it can support adaptation either with or without some labeled target domain instances. Experiment results on three NLP tasks show that while regular semi-supervised learning methods and supervised learning methods can be applied to domain adaptation without considering domain difference, they do not perform as well as our new method, which explicitly captures domain difference. Our results also show that incorporating and exploiting more information from the target domain is much more useful than excluding misleading training examples from the source domain. The framework opens up many interesting future research directions, especially those related to how to more accurately set/estimate those weighting parameters.

Acknowledgments

This work was in part supported by the National Science Foundation under award numbers 0425852 and 0428472. We thank the anonymous reviewers for their valuable comments.

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120–128.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proc. of EMNLP*, pages 285–292.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *J. Artificial Intelligence Res.*, 26:101–126.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of HLT-NAACL*, pages 1–8.

Y. Grandvalet and Y. Bengio. 2005. Semi-supervised learning by entropy minimization. In *NIPS*.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsupervised PCFG adaptatin to novel domains. In *Proc. of HLT-NAACL*, pages 126–133.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.