A Comparison of Methods for Transductive Transfer Learning

Andrew Arnold, Ramesh Nallapati and William W. Cohen
Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{aarnold, nmramesh, wcohen}@cs.cmu.edu

Abstract

In this paper we examine the problem of domain adaptation for protein name extraction. First we define the general problem of transfer learning and the particular subproblem of domain adaptation. We then describe some current state of the art supervised and transductive approaches involving support vector machines and maximum entropy models. Using these as inspiration, we turn to the unsupervised version of the problem and introduce a novel maximum entropy based technique, pseudo-label based rescaling (PLR), that achieves comparable performance with no labeled target data. We present the results of experimental comparisons between all the methods described and conclude with a discussion of trends observed and promising routes for future work.

1 Introduction

1.1 Problem formulation

Consider the task of *named entity extraction* (NER). Specifically, you are given a corpus of encyclopedia articles in which all the personal name mentions have been labeled. The standard supervised machine learning problem is to learn a classifier over this training data that will successfully label unseen test data drawn from the same distribution as the training data, where "same distribution" could mean anything from having the train and test articles written by the same author to having them written in the same language. Having successfully trained a named entity classifier on this encyclopedia data, now consider the problem of learning to classify tokens as names in instant messenger data. Clearly the problems of identifying names in encyclopedia articles and instant messages are closely related, and learning to do well on one should help your performance on the other. At the same time, however, there are serious differences between the two problems that need to be addressed. For instance capitalization, which will certainly be a useful feature in the encyclopedia problem, may prove less informative in the instant messenger data since the rules of capitalization are followed less strictly in that domain. Thus there seems to be some need for altering the classifier learned on the first problem (called the *source domain*) to fit the specifics of the second problem (called the *target domain*). This is the problem of *domain adaptation* and is considered a type of *transfer learning*.

The intuitive solution seems to be to simply train on the target domain data. Since this training data would be drawn from the same distribution as the data you will ultimately test over, this approach avoids the transfer issue entirely. The problem with this idea is that often large amounts of labeled data are not available in the target domain. It has been shown that even small amounts of labeled target data can greatly improve transfer results [6, 7]. There has been relatively little work, however, on the case when there is no labeled target data available, that is, totally unsupervised domain adaptation. This is the main problem we address in this paper.

One idea for approaching this unsupervised case is the use of maximum entropy classifiers. The maximum entropy model [15, 3] has proven very successful on many NER tasks. One of its strengths is its ability to take advantage of arbitrary, not necessarily independent features. During training, maximum entropy allocates certain weights to these features. Normally, in the non-transfer case, these weights are set so as to produce a model that maximizes the likelihood of the training data. Then, when the trained classifier is applied to unseen data (but data drawn from the same distribution as the training data) it will also give reasonable predictions. In the transfer setting, however, the distribution of the data is different between the train (source) and test (target) data. This is true for both the marginal distribution of the features, and more problematically, the conditional distribution of the class labels given the features. This means the weights assigned during training may not generalize well to test data drawn from the target domain. Our solution, presented in this paper, is to modify the way weights are assigned by maximum entropy. Instead of blindly maximizing the likelihood of the training data, we want to add some regularization that takes into account the distribution of the target domain. More specifically, we would like to use target domain data to down-weight features that might excel on the source data, but won't transfer to the target data. This will force maximum entropy to allocate that weight somewhere else, where it might not classify the training data as well, but will transfer better to the target data.

Since the target data is unlabeled, however, this poses a problem. We can estimate the marginal distribution of the target features, but not the conditional. Empirically we have seen that incorporating just the marginal information is not enough to improve performance. Thus we propose an EM-like algorithm for estimating a kind of pseudo-conditional distribution for the target data. This works by training a classifier on the source data, then using this classifier to get predicted class-probabilities for the target data. Using these predicted pseudo-labels, we can generate a target pseudo-conditional distribution. We then adjust the weights of the features in the source data to match this target conditional, and then train again on the adjusted data. The trade-off between setting the feature weights to match the source distribution and matching the target distribution can be controlled by a regularization parameter. This regularization is important because we believe the source data to be based on correct labels, but not quite relevant data, while the target data is based on faulty labels, but is at least drawn from the correct distribution.

1.2 Domain

We now turn to *protein name extraction*, an interesting problem domain in which to test these methods. In this setting you are given text related to biological research (usually abstracts, captions, and full body text from biological journal articles) which is known to contain mentions of protein names. The goal is to identify which words are part of a protein name mention, and which are not. Various machine learning methods and approaches have been applied to this particular problem [17, 22] with varying success. One major difficulty is that there is a large variance in how these proteins are mentioned and annotated between different authors, journals, and sub-disciplines of biology. There are even differences between protein name usage in different sections of the same article. For instance, the full name of a protein may be given in the introduction, but an abbreviation may be used in a subsequent caption. Because of this variance it is often difficult to collect a large corpus of truly identically distributed training examples. Instead, researchers are often faced with heterogeneous sources of data, both for training and testing, thus violating one of the key assumptions of most standard machine learning algorithms.

1.3 Previous work

Our present work draws on a long line of closely related research. One of the first formulations of the transfer learning problem was presented over 10 years ago by Thrun [20]. More recently there has been a focus on using source data to learn various types of priors for the target data [16]. Other techniques have tried to quantify the generalizability of certain features across domains [10, 11], or tried to exploit the common structure of related problems [2, 4].

Transfer learning is also closely related to *semi-supervised learning* [9, 23], in which one has access to both labeled and unlabeled data from the same domain at train time, and *transductive learning* [21, 12, 14, 18], in which one has access to the unlabeled testing data at training time (but still assumes the train and test distributions are the same).

Another related problem is *multi-task learning* [1, 19]. In this case, the distribution of the data does not change, but the task (and therefore the labels) do. For instance, a researcher may have previously learned to classify protein names, and now wants to learn to classify cell names. She might assume that there are many common features that would be informative for both tasks, but would need a way to find out which ones generalize across tasks, and which are task-dependent.

2 Transfer is not transduction

Given an example x and a class label y, the standard statistical classification task is to assign a probability, $p(y|x)$, to x of belonging to class y. In the binary classification case the labels are $Y = \{0, 1\}$. In the case we examine, the examples are represented as binary vectors where each value in the vector represents the presence or absence of a feature in a certain token t. That is, given the set of feature functions $\mathcal{F} = f_1...f_F$, where F is the number of features, and a token t_i drawn from the corpus $\mathcal{T} = t_1...t_N$, where N is the number of examples, we can construct the feature vector $x_i = \mathcal{F}(t_i) = \langle f_1(t_i), ..., f_F(t_i)\rangle$. Each $f_j(t_i)$ will be 1 if that feature function is true on the token, and 0 otherwise. We can further group these individual feature vectors into the set $X_{train} = x_1...x_N$, which we will call our training set. In order to evaluate our performance, we usually create another set X_{test}, formed in a similar way, upon which we want to use our trained classifier to make predictions.

Table 1. Summary of learning settings. Natural names are noted along with equivalences, where applicable. For all settings we assume $\mathcal{D}^{source}_{train}$ is used, and Y_{test} is unknown. Settings for which we have run experiments (see table 3) are marked in bold, along with their experimental name.

Natural name for learning setting	Experiment name	Auxiliary data		Test data	
		Domain	$Y_{auxiliary}$	Domain	X_{test}
classic supervised learning	SuprvNonTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{source}$	unseen
classic transductive learning	SuprvNonTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{source}$	seen
classic transfer learning	UnsuprvTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{target}$	unseen
transductive transfer learning	UnsuprvTransfer	$\mathcal{D}^{source}$	seen	$\mathcal{D}^{target}$	seen
classic semi-supervised learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{source}$	unseen
transductive semi-supervised learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{source}$	seen
semi-supervised transfer learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	unseen
transductive semi-supervised transfer learning		$\mathcal{D}^{source}$	unseen	$\mathcal{D}^{target}$	seen
reverse-transfer supervised learning[1]		$\mathcal{D}^{target}$	seen	$\mathcal{D}^{source}$	unseen
reverse-transfer transductive supervised learning[1]		$\mathcal{D}^{target}$	seen	$\mathcal{D}^{source}$	seen
supervised inductive transfer learning[2]	SuprvTransfer	$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	unseen
supervised transductive transfer learning[2]		$\mathcal{D}^{target}$	seen	$\mathcal{D}^{target}$	seen
reverse-transfer semi-supervised learning[1]		$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{source}$	unseen
reverse-transfer transductive semi-supervised learning[1]		$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{source}$	seen
unsupervised inductive transfer learning[2]		$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{target}$	unseen
unsupervised transductive transfer learning[2]	UnsuprvTransfer	$\mathcal{D}^{target}$	unseen	$\mathcal{D}^{target}$	seen

[1] These settings and names are unusual and not likely in practice, but are included for completeness.
[2] Equivalent to its classic version if we exclude the $\mathcal{D}^{source}_{train}$ data.

In the non-transfer learning problem X_{test} and X_{train} are both assumed to have been drawn from the same distribution, $\mathcal{D}$. In the transfer setting, however, we would like to apply our trained classifier to examples drawn from a distribution different from the one upon which it was trained. We therefore assume there are two different distributions, $\mathcal{D}^{source}$ and $\mathcal{D}^{target}$, from which data may be drawn. Given this notation we can then precisely state the transfer learning problem as trying to assign labels Y to test data X^{target}_{test} drawn from $\mathcal{D}^{target}$, given training data X^{source}_{train} drawn from $\mathcal{D}^{source}$.

In supervised inductive machine learning, X_{train} is known and labeled (that is Y_{train} is also known), while both X_{test} and Y_{test} are completely hidden. In the transductive case, however, X_{test} (but, importantly, not Y_{test}), is also known at training time. That is, the learning algorithm knows exactly which examples it will be evaluated on after training. This can be a great asset to the algorithm, allowing it shape its decision function to match and exploit the properties seen in X_{test}.

It is important to point out that transduction is orthogonal to transfer. That is, one can have a transductive algorithm that does or does not make the transfer learning assumption, and vice verse. Much of the work in this paper, however, was inspired by the belief that, although distinct, these problems are nevertheless intimately related. More specifically, when trying to solve a transfer problem between two domains, it seems intuitive that looking at the data of the target domain during training will improve performance over ignoring this source of information. Similarly, even if one believes he is not solving a transfer problem, it may still be beneficial to model one's training and test data as if they were not identically distributed.

Finally, there is the issue of supervision. We can define a third set of data, $X_{auxiliary}$, that is used, along with X_{train}, to train the classifier. In the case of non-transfer learning, if $X_{auxiliary}$ is labeled, this is just standard supervised learning (since X_{train} and $X_{auxiliary}$ are drawn from the same distribution). If $X_{auxiliary}$ is unlabeled, then this is semi-supervised learning. There is a similar distinction in the transfer case, where $X_{auxiliary}$ can be either labeled or unlabeled, and drawn from $\mathcal{D}^{source}$ or $\mathcal{D}^{target}$. Unfortunately, however, it does not seem possible to use labeled test data in the transductive transfer case. The reason is obvious: in transduction you can see your test data during training time. Thus, if X_{test} were labeled, there would be no need to train a classifier in the first place. These various conditions for performing each kind of learning are summarized in table 2.

Given this analysis then, it seems that one can do supervised inductive transfer learning, and unsupervised transductive transfer learning, but not supervised transductive transfer learning. In other words, in the transfer case, during training you have to choose whether you would rather know exactly what your unlabeled test data will look like or remain ignorant of this but instead see some labeled data drawn from the same distribution as your eventual test data.

3 Methods considered

3.1 Maximum entropy models

Entropy maximization (MaxEnt) is a way of modeling the joint distribution of examples and labels. Given a set of training examples $X_{train} \equiv \{x_{train_1}, \ldots, x_{train_N}\}$, their labels $Y_{train} \equiv \{y_{train_1}, \ldots, y_{train_N}\}$, and the set of features $\mathcal{F} \equiv \{f_1, \ldots, f_F\}$, MaxEnt learns a model consisting of a set of weights $\Lambda = \lambda_1 ... \lambda_F$ over the features so as to maximize the conditional likelihood of the training data, p($Y_{train}|X_{train}$), given the model p_Λ. In exponential parametric form, this conditional likelihood can be expressed as:

$$p_\Lambda(y_i|x_i) = \frac{1}{Z(x_i)} exp(\sum_j f_j(x_i, y_i)\lambda_j) \quad (1)$$

where Z is the normalization term:

$$Z(x_i) = \sum_{y' \in \{0,1\}} exp(\sum_j f_j(x_i, y')\lambda_j) \quad (2)$$

In order to avoid overfitting the training data, these λ's are often further constrained to be near 0 by the use of a regularization term which tries to minimize $||\Lambda||_1 \equiv \sum_j |\lambda_j|$. Thus the entire expression being optimized is:

$$\underset{\Lambda}{\text{argmax}} \quad \log \, p_\Lambda(Y|X) - \beta||\Lambda||_1 \quad (3)$$

where β is a parameter controlling the amount of regularization. Maximizing this likelihood is equivalent to constraining the conditional expectations of each feature in the learned model, $E_\Lambda[f_j|y]$, to match those found in the training data, which we denote $E_{train}[f_j|y]$:

$$E_{train}\left[f_j|y\right] = \frac{1}{N}\sum_i^N f_j(x_{train_i}, y_{train_i} = y) \quad (4)$$

where:

$$f_j(x_i, y_i = y) = \begin{cases} 1 & f_j \text{ occurs in example } x_i\text{, and } y_i = y \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Finally, we define E_Λ as:

$$E_\Lambda = \frac{1}{N}\sum_i \sum_y f_j(x_{train_i}, y) p\Lambda(y|x_{train_i}) \quad (6)$$

This model has proven highly successful and is used extensively in many domains including natural language and information extraction. One of its benefits is that the set of features $\mathcal{F}$ do not have to be independent of each other, which is not the case in some comparable methods. This frees the user to choose an arbitrary set of features she feels will best summarize the data without worrying about independence. One assumption the model does make is that the conditional expectation of the features is the same in the train and test data. This becomes a problem in the transfer learning setting and is addressed in the following sections.

3.1.1 Source trained prior models

One recently proposed method [6] for transfer learning in MaxEnt models involves modifying Λ's regularization term. First a model of the source domain, Λ^{source}, is learned by training on $\{X^{source}_{train}, Y^{source}_{train}\}$. Then a model of the target domain is trained over a limited set of labeled target data $\{X^{target}_{train}, Y^{target}_{train}\}$, but instead of regularizing this Λ^{target} to be near zero by minimizing $\sum_j |\lambda^{target}_j|$, Λ^{target} is instead regularized towards the previously learned source values Λ^{source} by minimizing $\sum_j |\lambda^{target}_j - \lambda^{source}_j|$. Thus the modified optimazation problem is:

$$\underset{\Lambda^{target}}{\text{argmax}} \quad \log \, p_{\Lambda^{target}}(Y|X) - \beta||\Lambda^{target} - \Lambda^{source}||_1 \quad (7)$$

It should be noted that this model requires Y^{target}_{train} in order to learn Λ^{target} and is therefore supervised transfer.

3.1.2 Feature space expansion

Another approach to the problem of supervised transfer learning is explored by Daumé [7, 10]. Here the idea is that there are certain features that are common between different domains, and others that are particular to one or the other. More specifically, we can redefine our feature set $\mathcal{F}$ as being composed of two distinct subsets $\mathcal{F}^{specific} \bigcup \mathcal{F}^{general}$, where the conditional distribution of the features in $\mathcal{F}^{specific}$ differ between X^{source} and X^{target}, while the features in $\mathcal{F}^{general}$ are identically distributed in the source and target. Given this assumption, there is an EM-like algorithm [10] for estimating the parameters of these distributions. There is also a simpler approach [7] of just making a duplicate copy of each feature in X^{source} and X^{target}, so whereas before you had $x_i = \langle f_1(x_i) ... f_F(x_i)\rangle$, you now have

$$\begin{aligned} x_i = \langle \; & f_1(x_i)^{specific}, f_1(x_i)^{general} \\ & ... f_F(x_i)^{specific}, f_F(x_i)^{general} \; \rangle \end{aligned} \quad (8)$$

where $specific$ is $source$ or $target$ respectively, and $f_j(x_i)^{specific}$ is just a copy of $f_j(x_i)^{general}$. Note that here

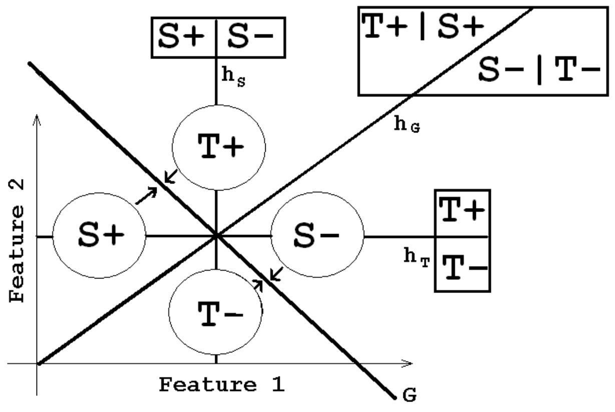

Figure 1. Illustration of feature space transformation in transfer learning problem. h_S and h_T easily separate the source and target data respectively, using only a single feature each. But a projection onto G is required before h_G can successfully separate both distributions at once.

we let $f_j(x_i)$ (the marginal form of equation 5) be 1 when f_j occurs in example x_i, and 0 otherwise. The idea is that by expanding the feature space in this way MaxEnt will be able to assign different weights to different versions of the same feature. If a feature is common in both domains its $general$ copy will get most of the weight, while its specific copies (f^{source} and f^{target}) will get less weight, and vice versa. Despite its simplicity, this method demonstrates an elegant interpretation of the transfer problem and works quite well

3.1.3 Conditional reweighting

The approach taken in this paper is slightly different. We attempt to directly modify the empirical conditional distribution of the features that MaxEnt tries to learn. For ease of notation we will allow $E^{source}[f_j|y]$ to mean $E_{x\in\mathcal{D}^{source}}[f_j(x)|y]$, and similarly for $target$. One problem with transfer in MaxEnt is that the conditional distribution of the features differs between the source and target domains. In other words, $E^{source}[f_j|y]$ does not necessarily equal $E^{target}[f_j|y]$. This becomes an issue because MaxEnt learns a model to maximize the likelihood of the training data based on these expectations. If the expectations in the train and test datasets are similar, then the Λ learned on the training data will also maximize the expectation of the test data. The more these distributions differ, however, the less well the trained model will perform.

Figure 1 illustrates this phenomenon. In this example, there are two features comprising the feature space. The distribution of the positive (+) and negative (-) classes of the source (S) and target (T) distributions are plotted with respect to these features. The supervised, non-transfer problems are simple in this setting since the source and target data are each easily separable in this feature space, by h_S and h_T respectively. For transfer learning, however, if we train on the source, we might learn the classifier h_S, which depends only *feature 1*. If we then attempt to classify the target data we will fail, since *feature 1* is a poor discriminator of the target data. What we would like to do is transform the feature space so that the distribution of the positive and negative classes in that transformed feature space is the same for both domains. This transformation is represented by G in the figure, a line upon which the data have been projected. Given this new transformation, h_G can easily be learned over the source data and subsequently performs equally well when transfered to the target data. Phrased in terms of maximum entropy, we are trying to learn a transformation $G()$ of the feature space $\mathcal{F}$ such that the conditional distributions of the source and target data are aligned:

$$E^{target}[f_j|y] = E^{source}[G(f_j)|y], \forall f_j \in \mathcal{F} \qquad (9)$$

The simple solution seems to be to ignore $E^{source}[f_j|y]$ and instead train our Λ based solely on $E^{target}[f_j|y]$. The problem with this, of course, is that in the unsupervised case we do not have Y^{target} and therefore cannot estimate $E^{target}[f_j|y]$. The best we can do is to estimate the target marginal $E^{target}[f_j]$. We should note, however, that training on the source data won't generally lead to $E^{source}[f_j|y] = E^{target}[f_j|y]$ since even the marginals of the two distributions might be different.

One way around the problem of missing conditionals is to assign *pseudo labels* $\hat{Y}^{target}$ to our target data. This would then allow us to estimate pseudo conditional expectations of the features in the target data $\hat{E}^{target}[f_j|\hat{y}]$. The question then becomes how best to assign these pseudo labels. One obvious approach is to use our classifier trained on the available, labeled source data to classify the unlabeled target data. We can then use the predicted target class labels to calculate the predicted target conditional expectations. With these $\hat{E}^{target}[f_j|\hat{y}]$'s we can do two things. Our first intuition might be to throw away our source data and proceed to directly train a new model $\hat{\Lambda}^{target}$. This is attractive because, ultimately, we are only concerned with testing on data drawn from $\mathcal{D}^{target}$ and so it seems that the sooner we can stop using the source data, the sooner we will stop being in a transfer problem, and so, hopefully, get better results by not violating our learner's assumptions. The downside to this, of course, is that this $\hat{\Lambda}^{target}$ would only be as good as the pseudo labels it was based on, which in turn are only as good as the classifier Λ^{source}. But Λ^{source} cannot be expected to perform very well on X^{target} since it was trained on only X^{source}. One idea for improving

$\hat{Y}^{target}$ (and thus $\hat{\Lambda}^{target}$) is to use our pseudo target conditional likelihoods, not to train our own target model, but instead to improve the source-trained model's performance on the target data. In this way, we will get better $\hat{\Lambda}^{target}$'s, yielding better $\hat{E}^{target}[f_j|\hat{y}]$'s, and so on. This is the idea behind our pseudo-label based rescaling (PLR) technique:

```
**Input**: $E^{source}[f_j|y]$, $\hat{E}^{target}[f_j|\hat{y}]$, $\theta$
**Output**: $E^{source}[f'_j|y]$
$f'_j|y = \theta * f_j + (1-\theta) * f_j * \frac{\hat{E}^{target}[f_j|\hat{y}]}{E^{source}[f_j|y]}$
$E^{source}[f'_j|y] = \frac{1}{N}\sum_i f_j(x_i, y_i = y)$
```

The effect is to rescale $f_j(x)$, putting more weight on features that occur frequently in the target but rarely in the source, and downweighting features that are common in the source but seldom seen in the target. Thus, after each iteration we get new, modified, source conditionals $E^{source}[f'_j|y]$ that we can use to train a new model and get a new Λ'^{source}. The parameter θ controls the degree to which we use the target pseudo conditionals to alter the the source conditionals. If $\theta = 1$ we ignore the pseudo conditionals, while setting $\theta = 0$ lets us reassign the source conditionals to match the target pseudo conditionals. We can see this by first observing that during each iteration the original features f_j are linearly transformed into modified f'_j:

$$f'_j = f_j * \frac{\hat{E}^{target}[f_j|\hat{y}]}{E^{source}[f_j|y]} \tag{10}$$

In effect this changes the originally binary-valued features into real-valued ones. The effect of this transformation can be written as:

$$\begin{aligned} E^{source}[f'_j|y] &= \frac{1}{N}\sum_{i \in source} f'_j(x_i, y_i) \\ &= \frac{\hat{E}^{target}[f_j|\hat{y}]}{E^{source}[f_j|y]} * \frac{1}{N}\sum_{i \in source} f_j(x_i, y_i) \\ &= \frac{\hat{E}^{target}[f_j|\hat{y}] * E^{source}[f_j|y]}{E^{source}[f_j|y]} \\ &= \hat{E}^{target}[f_j|\hat{y}] \end{aligned} \tag{11}$$

Thus, using the re-weighting of equation 10, we are able to express the conditional target distribution in terms of a transformed version of the source conditional, just as we had hoped in equation 9.

One final question is how, exactly, to calculate $\hat{E}^{target}[f_j|\hat{y}]$. If we trust our pseudo labels completely, we can just let $\hat{p}^{target}(y|x) = 1$ iff $y = \hat{y}$, and 0 otherwise. The problem with this is that our classifier may be more confident about its predicted labels for some examples than others. Specifically, we would assume it to do better on examples in the target domain that "looked like" examples in the source domain. Consequently, we would like to boost the contribution of these confident pseudo labels to the pseudo conditional expectation, while minimizing the contribution of less certain labels. Since MaxEnt outputs a conditional probability of each class label given an example, we already have a means for making this distinction. We have two options: we can ignore pseudo labels that have a predicted probability less than some threshold (maybe 90%), or we can weight the conditional expectation by this predicted class probability. We call the first method *thresholded hard labeling* (THL), and the second *soft labeling* (SL).

3.1.4 Biased thresholding

One further problem that arises in the transfer setting is that the marginal probability of a class label can differ significantly between the source and target domains. In the binary classification problem MaxEnt tries to estimate $p(y|x)$ for $y = \{0, 1\}$ (see equation 1). It then chooses the y with highest probability. The problem arises when $p^{source}(y)$ is very different from $p^{target}(y)$. In this case MaxEnt, having seen, say, very few positive examples in the source training data may develop a strong prior against positive examples when classifying the target data. One way around this is to adjust the decision rule used to decide which class label to predict. Naively, this threshold would be 50%, so that whichever class has the greater conditional probability, given the example and the model, would be predicted. If the model if over-predicting positive labels, however, we want to bias this threshold towards the negative examples.

More concretely, say we are training a classifier to predict whether it will rain tomorrow given today's weather report. Our training source data are weather reports from a city in a humid climate where the marginal chance of rain on any given day is 50%. We would like to apply our learned classifier to a target domain where the chance of rain is, on average, only 5%. In this case, we could expect a naively trained model to over-predict the chance of rain in the arid climate. To overcome this, we can adjust our threshold for predicting rain. Whereas previously we had predicted rain whenever our model told us there was a greater than 50% chance of rain, now we want to move this threshold up, so the model must be, say, 95% sure that it will rain, given its humid model, before we actually predict rain in the arid climate.

But how, exactly, do we set this threshold? One natural idea is to set the threshold so that the percentage of unlabeled target examples predicted to be in the positive class by the source-trained classifier is equal to some prior belief of the true proportion of positive examples in the target domain. Although this technically violates the terms of totally unsupervised transfer learning, in practice estimating this single parameter over the target domain does not require nearly as much labeled target data as learning all the parameters of a fully supervised model, and thus serves as a nice compromise between the two extremes. It should be noted that this kind of thresholding is not necessary when ranking is the goal instead of classification.

3.2 Support vector machines

Support vector machines (SVM's) [13] take a different approach to the binary classification problem. Instead of explicitly modeling the conditional distribution of the data and using these estimates to predict labels, SVM's try to model the data geometrically. Each example is represented as an F-dimensional real-valued vector of features and is then projected as a point in F-dimensional space. The algorithm then fits a discriminative hyperplane between the positively and negatively labeled training examples so as to best separate the two classes. This separation is called the margin, and thus SVM's belong to the margin based approach to classification. This simplification has proven very successful as SVM's currently have some of the lowest error rates of any popular learning algorithm. But this performance does come at a cost. In the transfer domain, for instance, we are acutely aware of our need for confidences in order to combine the source and target models. While SVM's do not have an explicit notion of probability, they do rely on the related concept of clustering. Specifically, the cluster assumption is that examples with common labels will lie near each other in feature space. Deciding how near, and in what feature space, are often difficult tasks in and of themselves, but can often be addressed by the thoughtful use of kernels.

3.2.1 Inductive SVM

As far as we know, there has not been much work on using inductive support vector machines for transfer learning. This may be due, in part, to the relative difficulty of interpreting and modifying the distances used in margin based models, as opposed to probabilistic ones. Since most transfer approaches rely on exploiting some connection between either the data or models trained in the source and target domains to exploit, the lack of an intuitive such connection in SVM's remains an obstacle. There has recently been an intriguing line of inquiry [18] into developing kernels and reproducing kernel Hilbert spaces that allow for this kind of communication between SVM models. They looked specifically at the non-transfer semi-supervised setting, but given their framework, it seems that a related transfer approach should be possible.

3.2.2 Transductive SVM

Transduction with SVM's, in contrast to probabilistic models, is quite intuitive. Whereas, in the inductive case, we tried to fit a hyperplane to best separate the labeled training data, in the transductive case, we add in unlabeled testing data which we must also separate. Since we do not know the labels of the testing data, however, we cannot perform a straight forward maximization, as in the supervised case. Instead, one can use an iterative algorithm [12] similar in flavor to the MaxEnt pseudo-label based rescaling (PLR) algorithm of section 3.1.3. Specifically, a hyperplane is trained on the labeled source data and then used to classify the unlabeled testing data. As in PLR, one can adjust how confident the hyperplane must be in its prediction in order to use a pseudo-label during the next phase of training (since there are no probabilities, large margin values are used as a measure of confidence). The pseudo-labeled testing data is then, in turn, incorporated in the next round of training. The idea is to iteratively adjust the hyperplane until it is very confident on most of the testing points, while still performing well on the labeled training points. In other words, we want to choose a hyperplane to maximize the margin over the training data and the pseudo-labeled testing data (this idea is closely related to the concept of entropy minimization outlined in [9]). To this end, the pseudo-labels of positive and negative testing examples can be switched in order to improve this margin metric over the whole dataset. As the algorithm proceeds the testing data's pseudo labels get more and more weight. One complication of this method for transductive SVM, however, is the time it takes to try the different possible labelings for the testing data. This issue can be addressed by using a spectral graph approach [14], thus allowing for faster convergence and a more general framework.

Although it does not specifically address the case of having different distributions for the labeled training and unlabeled testing data, using transductive SVM's for transfer problems does allow for the incorporation of some information from the target data's different marginal distribution and thus demonstrates improved performance over inductive SVM applied to the same problem.

3.2.3 Prior class probabilities and cost factors

As with the maximum entropy approaches described in section 3.1.4, transductive SVM's used for transfer can also suffer from a disparity between the prior proportion of positive examples in the source and target domains. In this case, a similar biasing technique can be employed to adjust the SVM's classification rule. Specifically, whereas the SVM usually just considers which side of the hyperplane a test example is on in determining its label (i.e., a threshold of 0), this threshold can be moved so that some points

Table 2. Summary of data used in experiments

Corpus name	Abstracts	Tokens	% Positive
UT	748	216,795	6.6%
Yapex	200	60,530	15.0%

that lie nearest on the negative side of the hyperplane and would normally be given a negative label, would instead receive a positive one, or vice verse. As with MaxEnt models, this relatively small piece of information (the proportion of positive examples in the target data) can effect a dramatic increase in performance.

Another related way to deal with a disproportionate number of positive or negative examples is to adjust the loss function minimized by SVM. Instead of treating false-positive and false-negative classification mistakes equally, if you, for example, expect a larger proportion of negative examples (as is often the case in named entity extraction), you might weigh false-negative errors more heavily (since each rare true-positive occurrence you miss is dear).

4 Investigation

4.1 Data

Our corpora of abstracts from biological journals come from two sources: University of Texas, Austin [5] and Yapex [8]. Each abstract was tokenized and each token was hand-labeled as either being part of a protein name or not. Some summary statistics for these data are shown in Table 2. We purposely chose corpora that differed in two important dimensions: the total amount of data collected and the relative proportion of positively labeled examples in each dataset. Specifically, UT has over three times as many tokens as Yapex but has only half the proportion of positively labeled protein names. This disparity is not uncommon in the domain and could be attributed to differing ways the data sources were collected and annotated. Specifically, since we are performing binary classification on the data, if the protein mention annotations in Yapex tend to be longer (that is, extend for more tokens) then the proportion of positively labeled tokens will be higher in Yapex. For all our experiments, we used the larger UT dataset as our source domain and the smaller Yapex dataset as our target.

4.2 Experiments and results

We assessed the relative performance of the various methods described in section 3 across the three dimensions of supervision, transduction and transfer (see table 1). Because of the relatively small proportion of positive examples in both the UT and Yapex datasets, we were more interested in achieving both high precision and recall instead of simply maximizing classification accuracy. Since we were dealing with binary, and not sequential classification, the definition of these measures was straightforward: a token was correctly classified if its predicted label (POS or NEG) matched its true label, and vice verse. We summarize the combined precision and recall of the various methods with the common F1 measure. Table 3 summarizes the results under the unsupervised and weakly supervised (prior knowledge of proportion of positive examples known) settings.

Notice how MaxEnt dominates the non-transfer experiment, achieving an F1 of 82% compared to TSVM's 73%. This should not be surprising given MaxEnt's suitability to classification in the natural language domain. Moving to the unsupervised transfer setting causes all three methods' performances to fall, but MaxEnt falls most sharply, causing it to lose its entire lead over TSVM. Again, this might be explained in light of MaxEnt's dependence on its feature set. TSVM is able to adjust its hyperplane in light of the transfer data, even though it is unlabeled, because it knows where these points lie relative to the labeled training points in feature space. MaxEnt, in contrast, only knows the marginal expectations of the features in the target domain (not the conditionals) and this is not enough to help it fully exploit its statistical modeling capabilities. In the weakly supervised setting, finally, where the target dataset is still unlabelled but all algorithms are told the expected proportion of positive examples, TSVM excels. Again, while MaxEnt is able to make significant use of this information, it seems TSVM does a better job leveraging the prior knowledge into better performance. Speculating a bit, this is most likely because of the difference in the ways these two algorithms adjust their thresholds to achieve the desired predicted positive classification rate. For TSVM, moving the hyperplane directly relates the proportion of positively labeled examples to the relative position of points in feature space. For MaxEnt, however, the relationship between the raw features and the decision rule is less clear, allowing more intervening factors to dilute the flow of information from prior to prediction.

Towards the bottom of table 3 we see the effect of our pseudo-label based rescaling algorithm (*PLR*, section 3.1.3) on MaxEnt's unsupervised transfer performance. We use a conservative value of θ (.95) to ease the transition from source- to target-based conditional feature expectations, and the *soft labeling* technique for determining our pseudo-conditional expectations. Indeed, as expected from our previous analysis, iteratively combining the pseudo conditional feature expectations in the target data with the true conditionals of the source data improves the overall performance

Table 3. Summary of % accuracy (Acc), precision (Prec), recall (Rec), and F1 for inductive SVM (ISVM), transductive SVM (TSVM), regular maximum entropy (MaxEnt), pseudo-label based rescaling MaxEnt (PLR), prior-based regularized MaxEnt (Regularize), and feature expansion MaxEnt (Expand) models under the conditions of supervised non-transfer, (SuprvNonTransfer), transfer with unlabeled target data, (UnsuprvTransfer), transfer with unlabeled target data but prior knowledge of % positive in target, (UnsuprvTransferPrior), and transfer with labeled auxiliary target data (SuprvTransfer).

Method	SuprvNonTransfer				UnsuprvTransfer				UnsuprvTransferPrior				SuprvTransfer			
	Acc	Prec	Rec	**F1**	Acc	Prec	Rec	**F1**	Acc	Prec	Rec	**F1**	Acc	Prec	Rec	**F1**
ISVM	92	78	58	**67**	90	86	40	**54**	90	86	40	**55**	92	86	52	**65**
TSVM	92	68	79	**73**	91	86	46	**60**	92	72	75	**73**	93	86	58	**70**
MaxEnt	95	85	78	**82**	89	75	42	**54**	90	65	68	**67**	91	81	54	**65**
PLR, 1 iter	-	-	-	-	79	41	90	**56**	-	-	-	-	-	-	-	-
PLR, 2 iters	-	-	-	-	82	45	86	**59**	-	-	-	-	-	-	-	-
Regularize	-	-	-	-	-	-	-	-	-	-	-	-	96	87	84	**85**
Expand	-	-	-	-	-	-	-	-	-	-	-	-	93	84	62	**72**

on the target data. It seems this method is bounded, however, by the quality of the initial pseudo labels generated by the source-trained classifier. Given a relatively poor initial classification, how can we bootstrap our way up to higher and higher performance? This is certainly a question worthy of future study.

Finally, the last rows of table 3 compare the performance of the two methods for supervised transfer learning: the prior-based regularized maximum entropy method (*Regularize*, described in section 3.1.1), and the feature expanding version (*Expand*, described in section 3.1.2). The algorithms were trained on the full labeled source data and 80% of the target training data, and then evaluated on the held-out 20% target testing data. We can see that both methods handily outperform the totally unsupervised transfer methods described in the second column of table 3, and for the most part outperform even the weakly supervised versions in column three. This should not be surprising given the fact that the supervised methods can actually see some labeled examples from the target domain and thus, in the case of MaxEnt, better estimate the conditional expectation of the features in the target data. Likewise, since they have access to labeled target data, they can also assess the proportion of positive examples and adjust their decision functions accordingly. What is more surprising, however, is the fact that these methods do not significantly outperform the supervised non-transfer methods described in the first column of table 3. This suggests that these supervised transfer methods are relying almost entirely on their labeled target data in order to train their classifiers, and are not making full use of the large amount of labeled source data. One might assume that having access to almost four times as much related data, in the form of the labeled source data, would significantly boost their ability to classify the target data (this is, after all, one of the stated goals of transfer learning). Dishearteningly, in this instance, this seems not to be the case. The regularized maximum entropy model does outperform[1] the non-transfer MaxEnt, but not by as much as might have been hoped for.

In order to measure how much these supervised transfer methods' explicit modeling of the transfer problem was responsible for their performance, we compared them to the baselines of ISVM, TSVM, and MaxEnt trained on a simple concatenation of the labeled source and target training data (first three rows of the *SuprvTransfer* column). Evaluated on X_{test}^{target}, these transfer-agnostic methods clearly benefited from the addition of labeled target data (as compared to column *UnsuprvTransfer*), yet still yielded consistently lower F1 than the transfer-aware *Regularize* and *Expand* methods, suggesting that the mere presence of labeled sets of both types (source and target) of data is not enough to account for transfer methods' superior results. Instead, it is the modeling of the different domains in the transfer problem, even in simple ways, that can provide the extra boost to performance.

5 Conclusions

These experiments and analysis have shed light on a number of important issues and considerations related to the problems of transduction and transfer learning.

We have seen that even a small amount of prior knowledge about the target domain can greatly improve performance in a transfer problem. In contrast, however, even

[1]*Regularize* has F1 of 85 vs. *MaxEnt*'s 82. Significance was determined by comparing the 99% binomial confidence intervals for each method's recall and precision.

large amounts of source data cannot overcome the advantage of having access to labeled data drawn from the test distribution.

We have also seen the degree to which pseudo-labeling based schemes (in both TSVM's margin-based model and our PLR's MaxEnt-based model) can improve performance by incorporating the unlabeled structure of the target domain. The magnitude of this improvement is closely tied to the respective method's ability to relate this information (whether manifested as position in feature space or marginal distribution in probability space) to the labeled source data it has available and its eventual decision function.

Finally, we have seen that probabilistic methods, like maximum entropy, seem better able to make use of the raw information available in the features provided, as long as those features faithfully represent the distribution of the data to be tested upon. But margin based methods, like transductive support vector machines, seem better able to adapt their decision surfaces in light of new data drawn from a different distribution.

It seems, then, that the best transfer methods may lie somewhere between the two, leveraging maximum entropy's precise probabilistic modeling with SVM's robust margin based approach.

6 Future work

Given the promising results of our MaxEnt based pseudo-label based rescaling methods, we would like to further investigate the theoretical properties of the PLR-type algorithms. In particular, it would be nice to be able to guarantee convergence. We have some intuition that there exists a convex formulation of the transfer problem in terms of the interaction between the source and target conditionals expectation that would allow for such a result. The fact that transductive SVM makes better use of prior information, when supplied, suggests there is also room to improve MaxEnt's ability to exploit the advantages of the transductive setting, perhpas by incorporating prior information directly into PLR's inner loop, or adjusting its soft-thresholding.

In terms of the named entity extraction application, we are also looking towards applying these techniques to the sequential, rather than just binary labeling problem. Most transfer learning results have emphasized the use of structure in relating the source and target domain, and it seems sequential classifiers like conditional random fields [19] would be better equipped to exploit this structure.

References

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. In *JMLR 6*, pages 1817 – 1853, 2005.

[2] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *NIPS 20*, Cambridge, MA, 2007. MIT Press.

[3] A. L. Berger, S. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[4] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, Sydney, Australia, 2006.

[5] R. Bunescu, R. Ge, R. Kate, E. Marcotte, R. Mooney, A. Ramani, and Y. Wong. Comparative experiments on learning information extractors for proteins and their interactions. In *Journal of Artificial Intelligence in Medicine*, 2004. Data from *ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/proteins.tar.gz*.

[6] C. Chelba and A. Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. In D. Lin and D. Wu, editors, *Proceedings of EMNLP 2004*, pages 285–292. ACL, 2004.

[7] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, 2007.

[8] K. Franzén, G. Eriksson, F. Olsson, L. Asker, P. Lidn, and J. Cöster. Protein names and how to find them. In *International Journal of Medical Informatics*, 2002. Data from *http://www.sics.se/humle/projects/prothalt/ yapex_text_collection.tar*.

[9] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *CAP*, Nice, France, 2005.

[10] H. D. III and D. Marcu. Domain adaptation for statistical classifiers. In *Journal of Artificial Intelligence Research 26*, pages 101–126, 2006.

[11] J. Jiang and C. Zhai. Exploiting domain structure for named entity recognition. In *Human Language Technology Conference*, pages 74 – 81, 2006.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML 16*, 1999.

[13] T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer, 2002.

[14] T. Joachims. Transductive learning via spectral graph partitioning, 2003.

[15] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification, 1999.

[16] R. Raina, A. Y. Ng, and D. Koller. Transfer learning by constructing informative priors. In *ICML 22*, 2006.

[17] L. Shi and F. Campagne. Building a protein name dictionary from full text: a machine learning term extraction approach. In *BMC Bioinformatics 6:88*, 2005.

[18] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *ICML*, pages 824–831. ACM, 2005.

[19] C. Sutton and A. McCallum. Composition of conditional random fields for transfer learning. In *HLT/EMLNLP*, 2005.

[20] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, volume 8, pages 640–646. MIT, 1996.

[21] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[22] R. C. Wang, A. Tomasic, R. E. Frederking, and W. W. Cohen. Learning to extract gene-protein names from weakly-labeled text in preparation. In *preparation*, 2006.

[23] X. Zhu. Semi-supervised learning literature survey. In *Technical Report 1530*. University of Wisconsin, 2005.