The Necessity of Combining Adaptation Methods

Ming-Wei Chang, Michael Connor and Dan Roth
University of Illinois at Urbana Champaign
Urbana, IL 61801
{mchang21,connor2,danr}@uiuc.edu

Abstract

Problems stemming from domain adaptation continue to plague the statistical natural language processing community. There has been continuing work trying to find general purpose algorithms to alleviate this problem. In this paper we argue that existing general purpose approaches usually only focus on one of two issues related to the difficulties faced by adaptation: 1) difference in base feature statistics or 2) task differences that can be detected with labeled data.

We argue that it is *necessary* to combine these two classes of adaptation algorithms, using evidence collected through theoretical analysis and simulated and real-world data experiments. We find that the combined approach often outperforms the individual adaptation approaches. By combining simple approaches from each class of adaptation algorithm, we achieve state-of-the-art results for both Named Entity Recognition adaptation task and the Preposition Sense Disambiguation adaptation task. Second, we also show that applying an adaptation algorithm that finds shared representation between domains often impacts the choice in adaptation algorithm that makes use of target labeled data.

1 Introduction

While recent advances in statistical modeling for natural language processing are exciting, the problem of domain adaptation remains a big challenge. It is widely known that a classifier trained on one domain (e.g. news domain) usually performs poorly on a different domain (e.g. medical domain) (Jiang and Zhai, 2007; Daumé III, 2007). The inability of current statistical models to handle multiple domains is one of the key obstacles hindering the progress of NLP.

Several general purpose algorithms have been proposed to address the domain adaptation problem: (Blitzer et al., 2006; Jiang and Zhai, 2007; Daumé III, 2007; Finkel and Manning, 2009). It is widely believed that the drop in performance of statistical models on new domains is due to the shift of the joint distribution of labels and examples, $P(Y, X)$, from domain to domain, where X represents the input space and Y represents the output space. In general, we can separate existing adaptation algorithms into two categories:

Focuses on $P(X)$ This type of adaptation algorithm attempts to resolve the difference between the feature space statistics of two domains. While many different techniques have been proposed, the common goal of these algorithms is to find a better shared representation that brings the source domain and the target domain closer. Often these algorithms do not use labeled examples in the target domain. The works (Blitzer et al., 2006; Huang and Yates, 2009) all belong to this category.

Focuses on $P(Y|X)$ These adaptation algorithms assume that there exists a small amount of labeled data for the target domain. Instead of training two weight vectors independently (one for source and the other for the target domain), these algorithms try to relate the source and target weight vectors. This is often achieved by using a special designed regularization term. The works (Chelba and Acero, 2004; Daumé III, 2007; Finkel and Manning, 2009) belong to this category.

It is important to give the definition of an adaptation *framework*. An adaptation framework is specified by the data/resources used and a specific learning algorithm. For example, a framework that used only source labeled examples and one that used both source and target labeled examples should be considered as two different frameworks, even though they might use exactly the same training algorithm. Note that the goal of a good adaptation framework is to perform well on the target domain and quite often we only need to change the data/resource used to increase the performance without changing the training algorithm. We refer to frameworks that do not use target labeled data and focus on $P(X)$ as **Unlabeled Adaptation Frameworks** and refer to frameworks that use algorithms that focus on $P(Y|X)$ as **Labeled Adaptation Frameworks**.

The major difference between unlabeled adaptation frameworks and labeled adaptation frameworks is the use of *target labeled examples*. Unlabeled adaptation frameworks do not use target labeled examples[1], while the labeled adaptation frameworks make use of *target labeled examples*. Under this definition, we consider that a model trained on both source and target labeled examples (later referred as **S**+**T**) is a labeled adaptation framework.

It is important to combine the labeled and unlabeled adaptation frameworks for two reasons:

- **Mutual Benefit**: We analyze these two types of frameworks and find that they address different adaptation issues. Therefore, it is often beneficial to apply them together.

- **Complex Interaction**: Another, probably more important issue, is that these two types of frameworks are *not* independent. Different representations will impact how much a labeled adaptation algorithm can transfer information between domains. Therefore, in order to have a clear picture of what is the best labeled adaptation framework, it is necessary to analyze these two domain adaptation frameworks together.

In this paper, we assume we have both a small amount of target labeled data and a large amount of unlabeled data so that we can perform both unlabeled and labeled adaptation. *The goal of our paper is to point out the* **necessity** *of applying these two adaptation frameworks together*. To the best of our knowledge, this is the first paper that both theoretically and empirically analyzes the interdependence between the impact of labeled and unlabeled adaptation frameworks.

The contribution of this paper is as follows:

- Propose a theoretical analysis of the "Frustratingly Easy" (FE) framework (Daumé III, 2007) (Section 3).

  The theoretical analysis shows that for FE to be effective the domains must already be "close". At some threshold of "closeness" it is better to switch from FE to just pool all training together as one domain.

- Demonstrate the complex interaction between unlabeled and labeled approaches (Section 4)

  We construct artificial experiments that demonstrate how applying unlabeled adaptation may impact the behavior of two labeled adaptation approaches.

- Empirically analyze the interaction on real datasets (Section 5).

  We show that in general combining both approaches on the tasks of preposition sense disambiguation and named entity recognition works better than either individual method. Our approach not only achieves state-of-the-art results on these two tasks but it also reveals something surprising – finding a better shared representation often makes a simple source+target approach the best adaptation framework in practice.

2 Two Adaptation Aspects: A Review

Why do we need two types of adaptation frameworks? First, unlabeled adaptation frameworks are necessary since many features only exist in one domain. Therefore, it is important to develop algorithms that find features which work across domains. On the other hand, labeled adaptation frameworks

[1]Note that we still use labeled data from source domain in an unlabeled adaptation framework.

are also required because we would like to take advantages of target labeled data. Even though different domains may have different definitions for labels (say in named entity recognition, specific definition of PER/LOC/ORG may change), labeled data should still be useful. We summarize these distinctions in Table 1.

While these two aspects of adaptation both saw significant progress in the past few years, little analysis has been done on the interaction between these two types of algorithms[2].

In order to have a deep analysis, it is necessary to choose specific adaptation algorithms for each aspect of adaptation framework. While we mainly conduct analysis on the algorithms we picked, we would like to point out that the necessity of combining these two types of adaptation algorithms has been largely ignored in the community.

As our example adaptation algorithms we selected:

Labeled adaptation: FE framework One of the most popular adaptation frameworks that requires the use of labeled target data is the "Frustratingly Easy" (FE) adaptation framework (Daumé III, 2007). However, why and when this framework works remains unclear in the NLP community. The FE framework can be viewed as an framework that extends the feature space, and it requires source and target labeled data to work. We denote n as the total number of features[3] and m is the number of the "domains", where one of the domains is the target domain. The FE framework creates a global weight vector in $\mathbb{R}^{n(m+1)}$, an extended space for all domains. The representation $\mathbf{x}$ of the t-th domain is mapped by $\Phi_t(\mathbf{x}) \in \mathbb{R}^{n(m+1)}$. In the extended space, the first n features consist of the "shared" block, which is always active across all tasks. The $(t+1)$-th block (the $(nt+1)$-th to the $(nt+n)$-th features) is a "specific" block, and is only active when extracting examples from the task t. More formally,

$$\Phi_t(\mathbf{x}) = \left[\underbrace{\mathbf{x}}_{\text{shared}} \quad \overbrace{\mathbf{0} \ldots \mathbf{0}}^{(t-1) \text{ blocks}} \quad \underbrace{\mathbf{x}}_{\text{specific}} \quad \overbrace{\mathbf{0} \ldots \mathbf{0}}^{(m-t) \text{ blocks}} \right]. \quad (1)$$

A *single* weight vector $\bar{\mathbf{w}}$ is obtained by training on the modified labeled data $\{y_i^t, \Phi_t(\mathbf{x}_i^t)\}_{t=1}^m$. Given that this framework only extends the feature space, in this paper, we also call it the *feature extension* framework (still called FE). We will see in Section 3 that this framework is equivalent to applying a regularization trick that bridges the source and the target domains. As it will become clear in Section 3, in fact, this framework is only effective when there is target labeled data and hence belongs to labeled adaptation frameworks.

Although FE framework is quite popular in the community, there are other even simpler labeled adaptation frameworks that allow the use of target labeled data. For example, one of the simplest frameworks is the **S+T** framework, which simply trains a single model on the pooled and unextended source and target training data.

Unlabeled adaptation: Adding cluster-like features Recall that unlabeled adaptation frameworks find the features that "work" across domain. In this paper, we find such features in two steps. First, we use word clusters generated from unlabeled text and/or third party resources that spans domains. Then, for every feature template that contains a word, we *append* another feature template that uses the word's cluster instead of the word itself. This technique is used in many recent works including dependency parsing and NER (Koo et al., 2008; Ratinov and Roth, 2009). Note that the unlabeled text need not come from the source or target domain. In fact, in this paper, we use clusters generated with the Reuters 1996 dataset, a superset of the CoNLL03 NER dataset (Koo et al., 2008; Liang, 2005). We adopt the Brown cluster algorithm to find the word cluster (Brown et al., 1992; Liang, 2005). We can use other resources to create clusters as well. For example, in the NER domain, we also include gazetteers[4] as an unlabeled cluster resource, which can bring the domains together quite effectively.

[2]Among the previously mentioned work, (Jiang and Zhai, 2007) is a special case given that it discusses both aspects of adaptation algorithms. However, little analysis on the interaction of the two aspects is discussed in that paper

[3]We assume that the number of features in each domain is equal.

[4]Our gazetteers comes from (Ratinov and Roth, 2009).

Framework	Labeled Data	Unlabeled Data	Common Approach
Unlabeled Adaptation (Focus on $P(X)$)	Source	Encompasses Source and Target. May use other third party resources (dictionaries, gazetteers, etc.).	Generate features that span domains using unlabeled data and/or third party resources.
Labeled Adaptation (Focus on $P(Y\|X)$)	Source and Target	None	Train classifier(s) using both source and target training data, relating the two.

Table 1: Comparison between two general adaptation frameworks discussed in this paper. Each framework is specified by its setting (data required) and its learning algorithm. Multiple previous adaptation approaches fit in one of either framework.

While other more complex algorithms (Ando and Zhang, 2005; Blitzer et al., 2006) for finding better shared representation (without using labeled target data) have been proposed, we find that using straightforward clustering features is quite effective in general.

3 Analysis of the FE Framework

In this section, we propose a simple yet informative analysis of the FE algorithm from the perspective of multi-task learning. *Note that we ignore the effect of unlabeled adaptation in this section, and focus on the analysis of the FE framework as a representative labeled adaptation framework.*

3.1 Mistake Bound Analysis

While (Daumé III, 2007) proposed this framework for adaptation, a very similar idea had been proposed in (Evgeniou and Pontil, 2004) as a novel regularization term for *multitask* learning with support vector machines. Assume that $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ are the weight vector for the first domain to the m-th domain, respectively. The baseline approach is to assume that each weight vector is independent. Assume that we adopt a SVM-like optimization problem that consider all m tasks, the baseline approach is equivalent to using the following regularization term in the objective function: $\sum_{t=1}^{m} \|\mathbf{w}_t\|^2$.

In (Evgeniou and Pontil, 2004; Daumé III, 2007), they assume that $\mathbf{w}_t = \mathbf{u} + \mathbf{v}_t$, for $t = 1, \ldots m$, where $\mathbf{v}_t$ is the specific weight vector for t-th domain and $\mathbf{u}$ is a *shared* weight vector across all domains. The new regularization term then becomes

$$\|\mathbf{u}\|^2 + \sum_{t=1}^{m} \|\mathbf{v}_t\|^2. \quad (2)$$

Note that these two regularization terms are different, given that the new regularization term makes $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ not independent anymore. It follows that

$$\mathbf{w}_t^T \mathbf{x} = (\mathbf{u} + \mathbf{v}_t)^T \mathbf{x} = \bar{\mathbf{w}}^T \Phi_t(\mathbf{x}),$$

where

$$\bar{\mathbf{w}}^T = \begin{bmatrix} \mathbf{u}^T & \mathbf{v}_1^T & \ldots & \mathbf{v}_m^T \end{bmatrix}.$$

and $\|\bar{\mathbf{w}}\|^2$ equals to Eq. (2). Therefore, we can think feature extension framework as a learning framework that adopts Eq. (2) as its regularization term.

The **FE** framework was in fact originally designed for the problem of multitask learning so in the following, we propose a simple mistake bound analysis based on the *multitask* setting, where we calculate the mistakes on *all* domains[5]. We focus on multitask setting for two reasons: 1) the analysis is very easy and intuitive, and 2) in Section 4.1, we empirically confirm that the analysis holds for the adaptation setting.

In the following, we assume that the training algorithm used in the FE framework is the online perceptron learning algorithm (Novikoff, 1963). This allows us to analyze the mistake bound of the **FE** framework with the perceptron algorithm. The bound can give us an insight on when and why one should adopt the **FE** framework. By using the standard mistake bound theorem (Novikoff, 1963), we show:

Theorem 1. *Let $\mathcal{D}_t$ be the labeled data of domain t. Assume that there exist $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ such that*

$$y\mathbf{w}_t^T \mathbf{x} \geq \mu, \forall (\mathbf{x}, y) \in \mathcal{D}_t,$$

and assume that $\max_{(\mathbf{x},y)\in\mathcal{D}_t} \|\mathbf{x}\| \leq R^2, \forall t = 1 \ldots m$. Then, the number of mistakes made with online perceptron training (Novikoff, 1963) and the

[5] In the adaptation setting, one generally only cares about the performance on the target domain.

FE framework is bounded by

$$\frac{2R^2}{\mu^2}(\sum_{t=1}^{m}\|\mathbf{w}_t\|^2 - \frac{\|\sum_{t=1}^{m}\mathbf{w}_t\|^2}{m+1}). \quad (3)$$

Proof. Define $\bar{\mathbf{w}}$ as a vector in $\mathbb{R}^{n(m+1)}$. We claim that there exists a set $S_{\bar{\mathbf{w}}}$ such that for all $\bar{\mathbf{w}} \in S_{\bar{\mathbf{w}}}$, $\bar{\mathbf{w}}^T\Phi_t(\mathbf{x}) = \mathbf{w}_t^T\mathbf{x}$ for any domain $t = 1\ldots m$. Note that $\Phi_t(x)$ is defined in Eq. (1). We can construct $S_{\bar{\mathbf{w}}}$ in the following way:

$$S_{\bar{\mathbf{w}}} = \{\begin{bmatrix} s & (\mathbf{w}_1 - s) & \ldots & (\mathbf{w}_m - s)\end{bmatrix} \mid s \in \mathbb{R}^n\},$$

where s is an arbitrary vector with n elements.

In order to obtain the best possible bound, we would like to find the most compressed weight vector in $S_{\bar{\mathbf{w}}}$, $\mathbf{w}^* = \min_{\bar{\mathbf{w}}\in S_{\bar{\mathbf{w}}}}\|\bar{\mathbf{w}}\|^2$.

The optimization problem has an analytical solution:

$$\|\mathbf{w}^*\|^2 = \sum_{t=1}^{m}\|\mathbf{w}_t\|^2 - \|\sum_{t=1}^{m}\mathbf{w}_t\|^2/(m+1).$$

The proof is completed by the standard mistake bound theorem and the following fact: $\max_{\mathbf{x}}\|\phi_t(\mathbf{x})\|^2 = 2\max_{\mathbf{x}}\|\mathbf{x}\|^2 \leq 2R^2$. □

3.2 Mistake Bound Comparison

In the following, we would like to explore under what circumstances the **FE** framework can work better than individual models and the **S+T** framework using Theorem 1. The analysis is done based on the assumption that all frameworks use the perceptron algorithm.

Before showing the bound analysis, note that the framework proposed by (Evgeniou and Pontil, 2004; Finkel and Manning, 2009) is a generalization over these three frameworks (**FE**, **S+T**, and the baseline)[6]. However, our goal in this paper is different: we try to provide a deep discussion on *when and why* one should use a particular framework.

Here, we compare the mistake bounds of the feature sharing framework to that of the baseline approach, which learns each task independently[7]. In order to make the comparison easier, we make some simplifying assumptions. First, we assume that the problem contains only two tasks, 1 and 2. We also assume that $\|\mathbf{w}_1\| = \|\mathbf{w}_2\| = a$. These assumptions greatly reduce the complexity of the analysis and can give us greater insight into the comparisons.

Following the assumptions and Theorem 1, the mistake bound for the FE frameworks is

$$4(2 - \cos(\mathbf{w}_1, \mathbf{w}_2))R^2a^2/(3\mu^2) \quad (4)$$

This line of analysis leads to interesting bound comparisons for two cases. In the first case, we assume that task 1 and task 2 are essentially the same. In the second, more common case, we assume that they are different.

First, when we know a priori that task 1 and task 2 are essentially the same, we can combine the training data from the two tasks and train them as a single task. Therefore, given that we do not need to expand the feature space, the number of mistakes is now bounded by R^2a^2/μ^2. Note that this bound is in fact better than (4) with $\cos(\mathbf{w}_1, \mathbf{w}_2) = 1$. Therefore, if we know a priori that these two tasks are the same, training a single model is better than using the feature shared approach.

In practice, it is often the case that the two tasks are not the same. In this case, the number of mistakes of an independent approach on *both* task 1 and 2 will be bounded by the summation of the mistake bounds of task 1 and task 2. Therefore, using the independent approach, the number of mistakes for the perceptron algorithm on *both* tasks is bounded by $2R^2a^2/\mu^2$. The following results can be obtained by directly comparing the two bounds,

Corollary 1. *Assume there exists* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *which separate* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *respectively with functional margin* μ*, and* $\|\mathbf{w}_1\| = \|\mathbf{w}_2\| = a$*. In this case: (4) will be smaller than the bound of individual approach,* $2R^2a^2/\mu^2$*, if and only if* $\cos(\mathbf{w}_1, \mathbf{w}_2) = (\mathbf{w}_1^T\mathbf{w}_2)/(\|\mathbf{w}_1\|\|\mathbf{w}_2\|) > \frac{1}{2}$.

If we assume that there is no difference in $P(X)$ between domains and hence we can treat $\cos(\mathbf{w}_1, \mathbf{w}_2)$ as the similarity between two tasks, the above argument suggests:

- If the two tasks are very different, the baseline approach (building two models) is better than **FE** and **S+T**.

[6]The framework proposed by (Evgeniou and Pontil, 2004; Finkel and Manning, 2009) is a generalization of Eq. (1). It allows the user to weight each block of features. If we put zero weight on the shared block, it becomes the baseline approach. On the other hand, if we put zero weight on all task-specific blocks, the framework becomes the **S+T** approach.

[7]Note that mistake bound results can be generalized to generalization bound results. See (Zhang, 2002).

- If the tasks are similar enough, **FE** is better than baseline and **S+T**.
- If the tasks are almost the same, **S+T** becomes better than **FE** and baseline.

In Section 4.1, we will evaluate whether these claims can be justified empirically.

4 Artificial Data Experiment Study

In this section we will present artificial experiments. We have two primary goals: 1) verifying the analysis proposed in Section 3, and 2) showing that the representation shift will impact the behavior of the **FE** algorithm. The second point will be verified again in the real world experiments in Section 5.

Data Generation In the following artificial experiments we experiment with domain adaptation by generating training and test data for two tasks, source and target, where we can control the difference between task definitions. The general procedure can be divided into two steps: 1) generating weight vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ (for source and target respectively), and 2) randomly generating labeled instances for training and testing using $\mathbf{z}_1$ and $\mathbf{z}_2$.

The different experiments start with the same basic $\mathbf{z}_1$ and $\mathbf{z}_2$, but then may alter these weights to introduce task dissimilarities or similarities. The basic $\mathbf{z}_1$ and $\mathbf{z}_2$ are both generated by a multivariate Gaussian distribution with mean $\mathbf{z}$ and a diagonal covariance matrix βI:

$$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{z}, \beta I), \mathbf{z}_2 \sim \mathcal{N}(\mathbf{z}, \beta I),$$

where $\mathcal{N}$ is the normal distribution and $\mathbf{z}$ is random vector with zero mean. Note that $\mathbf{z}$ is only used to generate $\mathbf{z}_1$ and $\mathbf{z}_2$. There is one parameter, β, that controls the variance of the Gaussian distribution. Hence we use β to roughly control the "angle" of $\mathbf{z}_1$ and $\mathbf{z}_2$. When β is close to zero, $\mathbf{z}_1$ and $\mathbf{z}_2$ will be very similar. On the other hand, when β is large, $\mathbf{z}_1$ and $\mathbf{z}_2$ can be very different. In these experiments, we vary β between 0.01 and 5 so that we are experimenting only with tasks where the weight the task difference is the "angle" or cosine between $\mathbf{z}_1$ and $\mathbf{z}_2$. Once we obtain the $\mathbf{z}_1$ and $\mathbf{z}_2$, we normalize them to the unit length.

After selecting $\mathbf{z}_1$ and $\mathbf{z}_2$, we then generate labeled instances $(\mathbf{x}, y)$ for the source task in the following way. For each example $\mathbf{x}$, we randomly generate n binary features, where each feature has 20% chance to be active. We then label the example by

$$y = sign(\mathbf{z}_1^T \mathbf{x}),$$

The data for the target task is generated similarly with $\mathbf{z}_2$. In these experiments, we fix the number of features n to be 500 and generate 100 source training examples and 40 target training examples, along with 1000 target testing examples. This matches the reasonable case in NLP where there are more features than training examples and each feature vector is sparse. In all of the experiments, we report the averaged testing error rate on the target testing data.

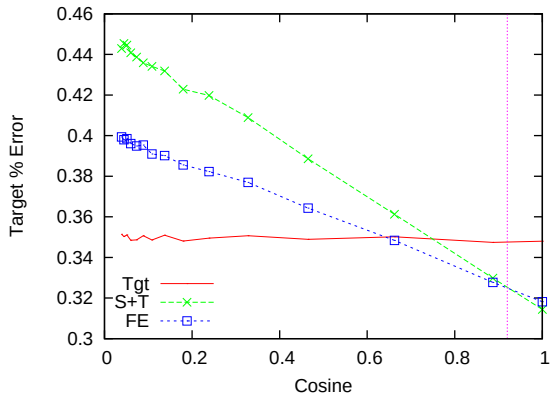
4.1 Experiment 1, FE algorithm

Goal The goal here is to verify our theoretical analysis in Section 3. Note that we do not introduce representation shift in this experiment and assume that both source and target domains use exactly the same features.

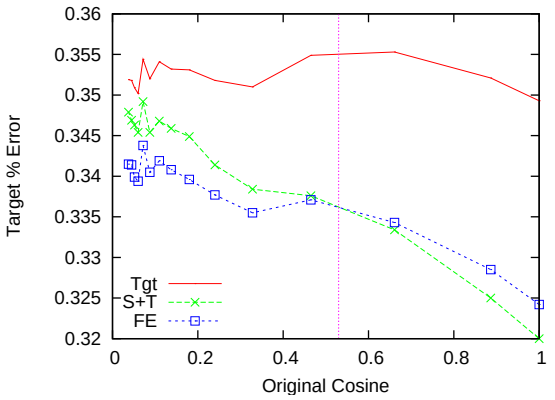
Result Figure 1(a) shows the performance of the three training algorithms as variance decreases and thus cosine between weight vectors (or measure of task similarity) goes to 1. Note that **FE** labeled adaptation framework beats **TGT** once the task cosine passes approximately 0.6. Initially FE slightly outperforms **S+T** until the tasks are close enough together that it is better to treat all the data as coming from one task. Note that while the experiments are based on the adaptation setting, the results match our analysis based on the multitask setting in Section 3.

4.2 Experiment 2, Unseen Features

Goal So far we have not considered the difference in $P(X)$ between domains. In the previous experiment, we used only cosine as our task similarity measurement to decide what is the best framework. However, task similarity should consider the difference in both $P(X)$ and $P(Y|X)$, and the cosine measurement is not sufficient for this. Here we construct a simple example to show that even a simple representation shift can change the behavior of the labeled adaptation framework. This case shows that **S+T** can be better than **FE** even when the tasks are not similar according to the cosine measurement.



(a) Basic Similarity



(b) Shared Features

Figure 1: Artificial Experiment comparing labeled adaptation performance vs. cosine between base weight vectors that defines two tasks, before and after cross-domain shared features are added. Figure (a) shows results from experiment 1. For **FE** adaptation algorithm to work the tasks need to be close (cosine > 0.6), and if the tasks are close enough (cosine ≈ 1, dividing line) then it is better to just pool source and target training data together (the **S+T** algorithm). Figure (b) shows results for experiment 3 when shared features are added to the base weight vectors as used in experiment 1. Here the cosine similarity measure is between the base task weight vectors before the shared features have been added. Both labeled adaptation algorithms effectively use the shared features to improve over just training on target. With shared features added the dividing line where **S+T** improves over **FE** decreases so even for tasks that are initially further apart, once clusters are added the **S+T** algorithm does better than **FE**. Each point represents the average of 2000 training runs with random initial $\mathbf{z}_1$ and $\mathbf{z}_2$ generating data.

Result The second experiment deals with the case where features may appear in only one domain but should be treated like known features in the other domain. An example of this are out of vocabulary words that may not exist in a small target training task, but have synonyms in the source training data. In this case if we had features grouping words (say by word meanings) then we would recover this cross-domain information. In this experiment we want to explore which adaptation algorithm performs best *before* these features are applied.

To simulate this case we start with similar weight vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ (sampled with variance = 0.00001, $\cos(\mathbf{z}_1,\mathbf{z}_2) \approx 1$), but then shift some set of dimensions so that they represent features that appear only in one domain.

$$\mathbf{z}_1 = (\mathbf{a}_1, \mathbf{b}_1) \rightarrow \mathbf{z}'_1 = (\mathbf{0}, \mathbf{b}_1, \mathbf{a}_1)$$
$$\mathbf{z}_2 = (\mathbf{a}_2, \mathbf{b}_2) \rightarrow \mathbf{z}'_2 = (\mathbf{a}_2, \mathbf{b}_2, \mathbf{0})$$

By changing the ratio of the size of the dissimilar subset $\mathbf{a}$ to the similar subset $\mathbf{b}$ we can make the two weight vectors $\mathbf{z}'_1$ and $\mathbf{z}'_2$ more or less similar. Using these two new weight vectors we can proceed as above, generating training and testing data.

Figure 2 shows the performance of the three algorithms on this data as the number of unrelated features are decreased. Over the entire range the combined algorithm **S+T** does better since it more efficiently exploits the shared similar $\mathbf{b}$ subset of the feature space. When the **FE** algorithm tries to create the shared features, it considers both the similar subset $\mathbf{b}$ and dissimilar subset $\mathbf{a}$. However, since $\mathbf{a}$ should not be shared, **FE** algorithm becomes less

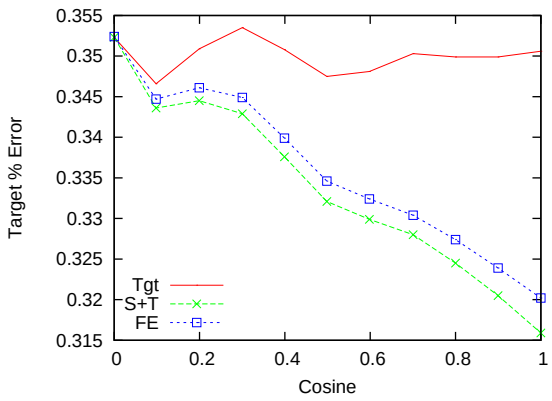


Figure 2: Artificial Experiment where unknown features are included in source or target domains, but not the other. The simple **S+T** adaptation framework is best able to exploit the set of shared features so performs best over the whole space of similarity in this setting.

effective than the **S**+**T** algorithm. See the bound comparison in Section 3.2 for more intuitions. With this experiment we have demonstrated that there is a need to consider label and unlabeled adaptation frameworks together.

4.3 Experiment 3, Shared Features

Goal A good unlabeled adaptation framework should try to find features that "work" across domains. However, it is not clear how these newly added features will impact the behavior of the labeled adaptation frameworks. In this experiment, we show that the new shared features will bring the domains together, and hence make **S**+**T** a very strong adaptation framework.

Result For the third experiment we start with the same setup as in the first experiment, but then augment the initial weight vector with additional shared weights. These shared weights correspond to the introduction of features that appear in both domains and have the same meaning relative to the tasks, the ideal result of unlabeled adaptation methods.

To generate this case we again start with $\mathbf{z}_1$ and $\mathbf{z}_2$ of varying similarity as in section 4.1, then generate a random weight vector for shared features and append this to both weight vectors.

$$\mathbf{z}_s \sim \mathcal{N}(0, I), \mathbf{z}_1'' = (\mathbf{z}_1, \gamma\mathbf{z}_s), \mathbf{z}_2'' = (\mathbf{z}_2, \gamma\mathbf{z}_s),$$

where γ is used to put increased importance on the shared weight vectors by increasing the total weight of that section relative to the base $\mathbf{z}_1$ and $\mathbf{z}_2$ subsets. In our experiments we use 100 shared features to the 500 base features and set γ to 2.

Figure 1(b) shows the performance of the labeled adaptation algorithms once shared features had been added. Here the x-axis is the cosine between the original task weight vectors, demonstrating how the shared features improve performance on potentially dissimilar tasks. Whereas in the first experiment **FE** does not improve over just training on target data until the cosine is greater than 0.6, once shared features have been added then both **FE** and **S**+**T** use these features to learn with originally dissimilar tasks. Furthermore the shared features tend to push the tasks 'closer' so that **S**+**T** improves over **FE** earlier. Comparing to Figure 1(a), there are regions where before shared features are added it is better to use **FE**, and after shared features are added it is better to use **S**+**T**. This shows that labeled adaptation and unlabeled are *not* independent. Therefore, it is important to combine these two aspects to see the real contribution of each adaptation framework.

In these three artificial experiments we have demonstrated cases where both **FE** or **S**+**T** are the best algorithm before and after representation changes like those created with unlabeled adaptation are imposed. This fact points to the perhaps obvious conclusion that there is not a single best adaptation algorithm, and the determination of specific best practices depends on task similarity (in both $P(X)$ and $P(Y|X)$), especially after being brought closer together with other adaptation approaches. If there is one common trend it is that often once two tasks have been brought close together using a shared representation, then the tasks are now close enough such that the simple **S**+**T** algorithm does well.

5 Real World Experiments

In Section 4, we have shown through artificial data experiments that labeled and unlabeled adaptation algorithms are not independent. In this section, we focus on experiments with real datasets.

For the labeled adaptation algorithms, we have the following options:

- **TGT**: Only uses target labeled training dataset.
- **FE**: Uses both labeled datasets.
- **FE**$^+$: Uses both labeled datasets. A modification of the FE algorithm, equivalent to multiplying the "shared" part of the FE feature vector (Eq. (1)) by 10 (Finkel and Manning, 2009).
- **S**+**T**: Uses both source and target labeled datasets to train a single model with all labeled data directly.

Throughout all of our experiments, we use SVMs trained with a modified java implementation[8] of LIBLINEAR as our underlying learning classifier (Hsieh et al., 2008). For the tasks that require structures, we model each individual decision using

[8] Our code is modified from the version available on `http://www.bwaldvogel.de/liblinear-java/`

Algorithm	**TGT**	**FE**	**FE$^+$**	**S+T**
SRC labeled data?	no	yes		
Target labeled data	Token F1			
(a) MUC7 Dev	58.6	70.5	**74.3**	73.1
(a) + cluster	77.5	82.5	**83.3**	**83.3**
(b) MUC7 Train	73.0	78.2	**80.1**	78.7
(b) + cluster	85.4	86.4	86.2	**86.5**

Table 2: **NER Experiments.** We bold face the best accuracy in a row and underline the runner up. Both unlabeled adaptation algorithms (adding cluster features) and labeled adaptation algorithm (using source labeled data) help the performance significantly. Moreover, adding cluster-like features also changes the behavior of the labeled adaptation algorithms. Note that after adding cluster features, **S+T** becomes quite competitive with (or slightly better than) the **FE$^+$** approach. The size of MUC7 develop set is roughly 20% of the size of the MUC7 training set.

a local SVM classifier then make our prediction using a greedy approach from left to right. While we could use a more complex model such as Conditional Random Field (Lafferty et al., 2001), as we will see later, our simple model generates state-of-the-art results for many tasks. Regarding parameter selection, we selected the SVM regularization parameter for the baseline model (**TGT**) and then fix it for all algorithms[9].

Named Entity Recognition Our first task is Named Entity Recognition (NER). The source domain is from the CoNLL03 shared task (Tjong Kim Sang and De Meulder, 2003) and the target domain is from the MUC7 dataset. The goal of this adaptation system is to maximize the performance on the test data of MUC7 dataset with CoNLL training data and (some) MUC7 labeled data. As an unlabeled adaptation method to address feature sparsity, we add cluster-like features based on the gazetteers and word clustering resources used in (Ratinov and Roth, 2009) to bridge the source and target domain. We experiment with both MUC development and training set as our target labeled sets.

The experimental results are in Table 2. First, notice that addressing the feature sparsity issue helps the performance significantly. Adding cluster-like features improves the Token-F1 by around 10%. On the other hand, adding target labeled data also helps the results significantly. Moreover, using both target labeled data and cluster-like shared representation are mutually beneficial in all cases.

Importantly, adding cluster-like features changes the behavior of the labeled adaptation algorithms. When the cluster-like features are not added, the **FE$^+$** algorithm is in general the best labeled adaptation framework. This result agrees with the results showed in (Finkel and Manning, 2009), where the authors show that **FE$^+$** is the best labeled adaptation framework in their settings. However, after adding the cluster-like features, the simple **S+T** approach becomes very competitive to both **FE** and **FE$^+$**. This matches our analysis in Section 4: *resolving features sparsity will change the behavior of labeled adaptation frameworks.*

We compare the simple **S+T** algorithm with cluster-like features to other published results on adapting from CoNLL dataset to MUC7 dataset in table 3. Past works on this setting often only focus on one class of adaption approach. For example, (Ratinov and Roth, 2009) only use the cluster-like features to address the feature sparsity problem, and (Finkel and Manning, 2009) only use target labeled data without using gazetteers and word-cluster information. Notice that because of combining two classes of adaption algorithms, our approach is significantly better than these two systems[10].

Preposition Sense Disambiguation We also test the combination of unlabeled and labeled adaption on the task of Preposition Sense Disambiguation. Here the data contains multiple prepositions where each preposition has many different senses. The goal is to predict the right sense for a given preposition in the testing data. The source domain is the SemEval 2007 preposition WSD Task and the target domain is from the dataset annotated in (Dahlmeier et al., 2009). Our feature design mainly comes from (Tratz and Hovy, 2009) (who do not evaluate their system on our target data). As our un-

[9]We use L2-hinge loss for all of the experiments, with $C = 2^{-4}$ for NER experiments and $C = 2^{-5}$ for the PSD experiments.

[10]The work (Ratinov and Roth, 2009) also combines their system with several document-level features. While it is possible to add these features in our system, we do not include any global features for the sake of simplicity. Note that our system is competitive to (Ratinov and Roth, 2009) even though our system does not use global features.

Systems	Cluster?	TGT?	P.F1	T.F1
Our NER	y	y	**84.1**	**86.5**
FM09	n	y	79.98	N/A
RR09	y	n	N/A	83.2
RR09 + global	y	n	N/A	86.2

Table 3: Comparisons between different NER systems. P.F1 and T.F1 represent the phrase-level and token-level F1 score, respectively. We use "Cluster?" to indicate if cluster features are used and use "TGT?" to indicate if target labeled data is used. Previous systems often only use one class of adaptation algorithms. Using both adaptation aspects makes our system perform significantly better than FM09 and RR09.

Algorithm	**TGT**	**FE**	**FE$^+$**	**S+T**
SRC labeled data?	no	yes		
Target labeled data	Accuracy			
10% Tgt	43.8	48.2	**51.3**	<u>49.7</u>
10% Tgt + Cluster	44.9	50.5	<u>51.8</u>	**52.0**
100% Tgt	59.5	<u>60.5</u>	60.3	**61.2**
100% Tgt + Cluster	61.3	<u>62.0</u>	61.2	**62.1**

Table 4: **Preposition Sense Disambiguation**. We mark the best accuracy in a row using the bold font and underline the runner up. Note that both adding cluster features and adding source labeled data help the performance significantly. Moreover, adding clusters also changes the behavior of the labeled adaptation algorithms.

labeled adaptation approach we augment all word based features with cluster information from separately generated hierarchical Brown clusters (Brown et al., 1992).

The experimental results are in Table 4. Note that we see phenomena similar to what happened in the NER experiments. First, both labeled and unlabeled adaptation improves the system. When only 10% of the target labeled data is used, the inclusion of the source labeled data helps significantly. When there is more labeled data, labeled and unlabeled adaption have similar impact. Again, using unlabeled adaption changes the behavior of the labeled adaption algorithms.

In Table 5, we compare our system to (Dahlmeier et al., 2009), who do not use the SemEval data but jointly train their preposition sense disambiguation system with a semantic role labeling system. With both labeled and unlabeled adaption, our system is significantly better.

Systems	ACC
Our PSD (**S+T** and cluster)	62.1
DNS09	56.5
DNS09 + SRL	58.8

Table 5: Comparison between different PSD systems. Note that after adding cluster features and source labeled data with **S+T** approach, our system outperforms the state-of-the-art system proposed in (Dahlmeier et al., 2009), even though they jointly learn a PSD and SRL system together.

6 Conclusion

In this paper, we point out the necessities of combining labeled and unlabeled adaptation algorithms. We analyzed the FE algorithm both theoretically and empirically, demonstrating that it requires both a minimal amount of task similarity to work, and past a certain level of similarity other, simpler approaches are better. More importantly, through artificial data experiments we found that applying unlabeled adaptation algorithms may change the behavior of labeled adaptation algorithms as representations change, and hence affect the choice of labeled adaptation algorithm. Experiments with real-world datasets confirmed that combinations of both adaptation methods provide the best results, often allowing the use of simple labeled adaptation approaches. In the future, we hope to develop a joint algorithm which addresses both labeled and unlabeled adaptation at the same time.

Acknowledgment We thank Vivek Srikumar for providing the baseline implementation of preposition sense disambiguation. We also thank anonymous reviewers for their useful comments. University of Illinois gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In Dekang Lin and Dekai Wu, editors, *EMNLP*.

D. Dahlmeier, H. T. Ng, and T. Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *EMNLP*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.

T. Evgeniou and M. Pontil. 2004. Regularized multi–task learning. In *KDD*.

J. R. Finkel and C. D. Manning. 2009. Hierarchical bayesian domain adaptation. In *NAACL*.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML*.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling. In *ACL*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

P. Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

A. Novikoff. 1963. On convergence proofs for perceptrons. In *Proceeding of the Symposium on the Mathematical Theory of Automata*.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*.

S. Tratz and D. Hovy. 2009. Disambiguation of preposition sense using linguistically motivated features. In *NAACL*.

Tong Zhang. 2002. Covering number bounds of certain regularized linear function classes. *J. Mach. Learn. Res.*