Literature Survey: Domain Adaptation Algorithms for Natural Language Processing

Qi Li
qli@gc.cuny.edu

Department of *Computer Science*
The Graduate Center, The City University of New York

June 13, 2012

Contents

Abstract

Traditional supervised learning algorithms assume that the training data and the test data are drawn from the same distribution. Models that are purely trained from training data are applied directly to the test data. Unfortunately, in practice this assumption is often too strong. Consider a common situation in which a large amount of manually labeled data is available in one domain, but the task is to make predictions on examples from a new domain with little or no labeled data. Given that the instances in the two domains are drawn from different distributions, traditional supervised learning can not achieve high performance on the new domain. For example, in the part-of-speech tagging task, the word *monitor* is likely to be a verb in the finance domain, while in the corpus of computer hardware, it's more likely to be a noun. Therefore domain adaptation algorithms are designed to bridge the distribution gap between the training data and the test data.

In this literature survey, we study the motivations, challenges and recently developed algorithms of domain adaptation in the field of natural language processing (NLP). We first categorize recent work into three classes according to their methodologies, and then review and compare some preventative algorithms in each category. Finally we summarize popular NLP applications involve domain adaptation.

1 Introduction

The notion of domain adaptation is closely related to transfer learning. Transfer learning is a general term that refers to a class of machine learning problems that involve different tasks or domains. In the literature, there isn't yet a standard definition of transfer learning. In some papers it's interchangeable with domain adaptation. Pan and Yang (2010) is a comprehensive survey of transfer learning techniques, which provided a clear definition of transfer learning:

> Given a source domain $\mathcal{D}_S$ and its learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and its learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function f_T in $\mathcal{D}_T$ using the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$.

If a learning problem also aims to improve the performance on $\mathcal{T}_S$ simultaneously, then it's usually called multi-task learning. Given different settings of $\mathcal{D}_S$, $\mathcal{T}_S$, $\mathcal{D}_T$ and $\mathcal{T}_T$, transfer learning can be roughly divided into two categories.

1. *Inductive transfer learning*, where $\mathcal{T}_T$ is different from $\mathcal{T}_S$.

2. *Transductive transfer learning*, where $\mathcal{T}_T$ and $\mathcal{T}_S$ are the same.

In each category, there are several different cases on the availability of labeled data:

1. Labeled data in $\mathcal{D}_S$ is available, but in $\mathcal{D}_T$ is not available.

2. Labeled data in both domains is available, but the amount of labeled data in $\mathcal{D}_T$ is small.

3. Labeled data in $\mathcal{D}_S$ is not available, but in $\mathcal{D}_T$ is available.

4. Labeled data in both domain are not available.

The first two cases are the most common situations, since in many applications it is very expensive to obtain labeled data in a new domain, while there is large amount of data available in some old domains. For example, manual annotations for part of speech tagging, syntactic parsing, named entity recognition, relation and event detection are available for news documents in English and other high-density languages, such as Tree Bank[1], Automatic Content Extraction (ACE) corpus[2] and CoNLL Named Entity Recognition corpus[3]. However, such training data is rarely available for other genres and domains such as emails, web forums and biomedical papers.

In this situation, If $\mathcal{T}_T$ and $\mathcal{T}_S$ are the same, then the learning task is often referred as domain adaptation. For example, consider we have a large collection of training data for parsing in news documents, if we want to obtain high performance parsing model for emails, gathering labeled data for emails is costly. Fortunately we can design algorithms to transfer knowledge from labeled data in news domain to the new email domain. Such techniques can relax the necessity of expensive manual annotation on the new data while keep high performance.

In this survey, we mainly focus on domain adaptation on natural language processing, which includes tasks such as part-of-speech tagging, named entity recognition, machine translation and sentiment analysis.

1.1 Notations

In this survey, we assume that training data is $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^n$, where each $x^i \in \mathcal{X}$ is an observed variable, and $y^i \in \mathcal{Y}$ is the output label/class of x^i. We

[1]http://www.cis.upenn.edu/ treebank/

[2]http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2006T06

[3]http://www.cnts.ua.ac.be/conll2003/ner/

use subscript S and T to distinguish source domain and target domain respectively. Therefore $\mathcal{D}_S$ means training data in the source domain, and $\mathcal{D}_T$ stands for training data in the target domain. In some cases, we also use subscript l and u to distinguish labeled and unlabeled data, for example $\mathcal{D}_{t,l}$ refers to labeled data in the target domain. Throughout this paper, we assume θ refers to parameters to any machine learning model we will discuss. Therefore $p(y|x;\theta)$ means the conditional probability of y given x under the model parameters θ.

1.2 Definition of Domain

In the literature, many definitions of the source domain and the target domain are proposed. In this survey, we restrict the notation of domain to some particular settings which are common in natural language processing tasks:

- Adaptation between different corpora: in this setting, each corpus can be considered as a unique domain. For example, we can consider ACE corpus as the source domain and CoNLL corpus as the target domain to perform named entity recognition (Finkel and Manning, 2009).
- Adaptation from a general dataset to a specific dataset: in this setting, the general dataset which consists of different corpora can be considered as source domain, while the specific dataset is considered as target domain. For example, in (Amittai et al., 2011)'s domain adaptation for machine translation, a combination of various available Chinese-English parallel corpora is considered as the source domain. The MT system developed on this source is then adapted to the Chinese-English task in the International Workshop on Spoken Language Translation (IWSLT).

- Adaptation between subtopics in the same corpus: in this setting, each subset of the whole corpus is considered as a domain. For example, (Wu et al., 2009) manually classified the ACE corpus into several topics, and treated the documents in each topic as one domain.
- Cross-lingual adaptation: if we have training data in one language, but want to build model for another language, then the datasets in each domain can be considered as one domain. For example, (Xu, Xu, and Wang, 2011) used English sentiment analysis corpus as the source domain to build the model for Chinese test data.

There are many other settings of domain adaptation, for example, image classification based on text, and Wifi localization in different time periods or mobile devices (Pan and Yang, 2010). They are beyond the scope of this survey.

1.3 Overview

Jiang and Zhai (2007a) analyzed some general properties of the domain adaptation problem in discriminative models from the theoretical point of view. Let $\mathcal{X}$ be the feature space for observed data, and $\mathcal{Y}$ be the set of output labels. In the fully supervised setting, each training instance (x, y) is drawn from a joint distribution $p(x, y)$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. $p(x, y)$ can be represented as $p(y|x) \cdot p(x)$. The objective of discriminative learning is to estimate model parameters θ in order to recover the conditional distribution $p(y|x)$, while common generative learning aims to recover the joint distribution $p(x, y)$. After the training process, given an unseen test example x_i, the model can predict its output label y by evaluating $arg \max_{y} p(y|x; \theta)$.

In the maximum likelihood estimation (MLE) framework, the objective of the learning problem can be interpreted as the following process: first, from the space of $\mathcal{X}$, the samples (x, y) are drawn from the joint distribution $p(x, y)$. And $p(x, y)$ can further be decomposed as $p(x) \cdot p(y|x)$. With the instances (x, y) from the true distribution, the learning algorithm defines the model distribution $\log p(y|x; \theta)$ to predict y given x. The model parameters θ are then estimated to maximize the log-likelihood of the whole sampled data set. This process can be generalized to generative learning easily, in this case, joint distribution $p(x, y; \theta)$ is considered instead of the conditional distribution $p(y|x; \theta)$.

In practice, since the real distribution $p(x, y)$ is unknown, the empirical estimation $\tilde{p}(x, y)$ from the sampled training data is used for approximation. In the fully supervised learning, the basic assumption is that if we have large amount of training data, then $\tilde{p}(x, y)$ can be a fairly good estimation of the real joint distribution $p(x, y)$. In contrast, in the case of domain adaptation, the training data and the test data are drawn from different underlying distributions, i.e. $p_S(x, y) \neq p_T(x, y)$. As a result, the above assumption doesn't hold. Therefore the challenge becomes how to approximate $p_T(x, y)$ instead of just using training data in D_S or even a small amount of training data in the target domain $D_{T,l}$.

Given this analysis, there are three general types of domain adaptation algorithms in the previous work. The first class of algorithms we will discuss in this survey is based on feature transformation (Daumé III, 2007; Blitzer, McDonald, and Pereira, 2006; Jiang and Zhai, 2007b; Guo et al., 2009; Xue et al., 2008). In this class, the assumption is that $p_T(y|x)$ differs from $p_S(y|x)$, but there exist some general features $x_g \in \mathcal{X}$ which have identical or similar conditional distributions in both source and target domains. Even when such features are very few or don't exist in the original feature space $\mathcal{X}$, there is still some correspondence between $p_S(y|x)$ and $p_T(y|x)$, which means it's possible to project the original feature space $\mathcal{X}$ into a new space $\mathcal{X}_{Tran}$ by using some projection methods $\theta \cdot \mathcal{X}$. Let's consider domain adaptation for part-of-speech tagging from the source domain *Wall Street Journal* to the target domain *Biomedical Abstract* (Blitzer, McDonald, and Pereira, 2006), the part-of-speech tags of some words differ in two domains either because they have different meanings in the two domains or one feature occurs in one domain but rarely occurs in another domain. For instance, the the noun *investment* only occurs in the *Wall Street Journal*. In contrast, some features are general in both domains. For instance, the verb *required* and the prepositions *from* and *for* have same meanings in the two domains. To illustrate this general picture, we show an example from Blitzer, McDonald, and Pereira (2006) in Table 1.

Biomedical	WSJ
the **signal** *required* to	**investment** *required*
stimulatory **signal** *from*	**buyouts** *from* buyers
essential **signal** *for*	to **jail** *for* violating

Table 1: Example of general and domain-specific features (Blitzer et al., 06) italicized are general features and bold are domain-specific features

However, just using these obvious general features is inadequate. There are two main challenges in such algorithms: (i) how to distinguish domain-specific features and general features (ii) how to find a new feature space $\mathcal{X}^{Trans}$ to encode the correspondence between source and target domains. To address these problems, some algorithms are proposed such as *correspond structural learning* (Blitzer, McDonald, and Pereira, 2006) and *Topic modeling* (Guo et al., 2009; Xue et al., 2008).

We will discuss these algorithms in Section 2.1.

The second class of algorithms exploit model priors to overcome the distribution gap between two domains (Finkel and Manning, 2009; Chelba and Acero, 2006; Chan and Ng, 2006). In discriminative learning, Gaussian prior is often used for regularization. Parameters θ is considered to be governed by the prior distribution $P(\theta)$. In order to approximate $p_S(y|x)$ from a large amount of source training data and a small set of target domain training data, adjusting the prior distribution $P(\theta)$ to some appropriate value can help produce a reasonable $p_S(y|x;\theta)$. This direction was studied in (Finkel and Manning, 2009; Chelba and Acero, 2006). In generative learning, the notation of *prior* refers to the prior distribution of an output label $p(y)$. One assumption is that the conditional distribution $p(x|y)$ is the same or similar in both domains, the gap between posteriori distribution $p(y|x)$ mainly comes from the difference between $p_S(y)$ and $p_T(y)$. Therefore a good estimation $p_T(y)$ based on the available data can boost the performance. Chan and Ng (2006) studied this direction in Naive Bayes framework.

Different from the above two classes, the third class of algorithms focus on instance level without changing the underlying machine learning models (Amittai et al., 2011; Xu, Xu, and Wang, 2011; Jiang and Zhai, 2007a). This type of algorithms is closely related to common semi-supervised learning framework such as self-training. The general idea is as follows: by weighting and selecting training instances, one can make $p(y|x;\theta)$ close to $p_T(y|x)$ by down-weighting or discarding instances which have different $p_S(y|x)$ and $p_L(y|x)$. In another case, assume $p(y|x)$ stays similar but $p(x)$ differs sharply in two domains, the resulting model $p(y|x;\theta)$ is highly influenced by the empirical distribution $\tilde{p}(x)$, then weighting and selecting training instances can recover such impact. The main challenge in this class is

Class	Level	Dependence of the underlying classifiers	Domain Gap	Extendability to Multiple Domains Learning
Feature space transformation (FST)	Feature level	Moderate	Large	Strong
Prior based adaptation (PBA)	Model level	Strong	Large	Strong
Instance selection/weighting (ISW)	Instance level	Weak	Small	Weak

Table 2: Comparison of three classes of algorithms

how to determine the instance weighting parameters or select which instances for training. Note that instance selection can be viewed as a special case of instance weighting in which the weighting parameters are binary-valued, i.e. either 1, which means to keep the instance, or 0, which means to discard the instance.

1.4 General Comparison

Table 2 compares some general aspects of these three categories. Please note that the conclusion is general for each class, but maybe not applicable to some particular algorithms. The first aspect is the level of adaptation: FST approaches are classified as feature level methods, since they studied the problem of designing new feature space to reduce the domain gap. PBA algorithms operate on the model level, in which we need to change the underlying model itself so that priors can be utilized for adaptation. ISW works on the instance level, since the algorithms in this class only focus on selecting and weighting instances without changing feature space or basic classifiers. The second aspect is the dependence of the underlying classifiers. If an adaptation method is more dependent on the underlying classifier, then it's less adaptable to other machine learning tasks. ISW is the most independent of the underlying classifiers, in theory, after the instance selection and weighting, any classifier can be applied, therefore these methods can be easily applied to complicated NLP tasks such as Machine Translation. PBA is most dependent of the underlying classifiers, these methods can not easily be adapted to new classifiers. The third aspect is the domain gap. Here domain gap means the difference between source and target domain. ISW is the most restrictive method in this aspect, it's suitable to the situation in which there exists many examples x_i in the source domain for which $p_S(y|x_i)$ is similar to $p_T(y|x_i)$. The fourth aspect is the ability of the method to deal with the case where multiple domains are available. In most cases, there is only one source domain and one particular target domain, in practice, multiple source domains can be available for adaptation. ISW methods can be hardly extended to this scenario, but many algorithms in FST and PBA are highly applicable to this case.

The remaining part of this paper is organized as follows: In Section 2, we review and critique some representative domain adaptation algorithms in each category. In Section 3, we summarize the common NLP applications of domain adaptation, and compare the different aspects and datasets in each task.

2 Domain Adaptation Algorithms for NLP

In this section, we will study the three categories of adaptation algorithms in detail. Section 2.1 discusses adaptation based on feature space transformation. Section 2.2 reviews methods based on changing priors, Finally Section 2.3 describes instance level adaptation approaches based on instance selection and weighting.

2.1 Feature Space Transformation

Changing the feature space or distinguishing subset of features can alleviate the gap between $p_S(y|x)$ and $p_T(y|x)$. The challenge is how to transfer original feature space $\mathcal{X}$ to a new space $\mathcal{X}'$ which is predictive for the target domain, and how to distill features that are predicative for both domains.

2.1.1 Simple Feature Augmentation

Daumé III (2007) introduced a very simple but interesting method, which can be considered as preprocessing for any statistical learning algorithm. In this approach, assume there is a large set of training data in the source domain, and a few labeled instances in the target domain. Each feature x_i in the source domain is duplicated to three versions: $< x_i, x_i, 0 >$. The first one is source domain specific version, the second one refers to domain independent version, and the last one means target domain specific version. Since x_i is from the source domain, the last column in this example is 0. Correspondingly, each feature x_i in the target domain is replicated as $< 0, x_i, x_i >$. After this simple step, the new feature vectors are fed into common supervised training algorithm. This method is extremely simple but the experiment results are very promising. The author gave an intuitive justification for this method. Consider the part-of-speech tagging task:

if a high weight is assigned to the domain-independent version, it means the feature behaves similarly in both domains. For example, the word "*the*" is commonly used as *determiner*, so the indicator feature of *determiner* should attain high weight for domain-independent version. On the other hand, if a high weight is assigned to the domain-specific feature, it's likely that this feature only occur frequently at this domain. For example, the word *monitor* is more likely to be a *none* rather than *verb* in *Computer Hardware* domain, then the indicator feature of *none* for this domain is very likely to attain high weight.

The most appealing advantage of this method is that it's very easy to implement, it just needs several lines of codes to duplicate the feature vectors. In addition, this simple method can be easily applied to multiple domains: each feature is duplicated for each domain plus a general version. Finally, it can be used with any statistical classifiers, since the feature duplication doesn't rely on any knowledge from the classifier. However, it has some limitations. First, it requires training data in the target domain, and the performance highly relies on the size of the target domain training data. It restricts the method to the cases where reasonable size of target domain training data is available. Second, this method is unable to deal with the case where domain-specific features are different but have some hidden correlations. Later we will compare some methods such as (Blitzer, McDonald, and Pereira, 2006) and (Guo et al., 2009) which exploit such correlations. Finally, this approach doesn't make formal distinction and connection between regularization terms for domain-specific and domain-independent features. The domain-independent feature weights and the domain-independent feature weights are governed by the same Gaussian prior. In later section we will show that this approach is a special case of Hierarchical Bayesian model (Finkel and Manning, 2009), which allows weights of domain-specific features and domain-independent features to have different priors.

2.1.2 Structural Correspondence Learning

Daumé III (2007)'s method simply duplicates each feature as domain-independent and domain-specific versions, it's unable to capture the potential correspondence if the context features in two domains are different. Blitzer, McDonald, and Pereira (2006) proposed Structural Correspondence Learning (SCL) for domain adaptation. SCL is borrowed from a multi-task learning algorithm called Structural learning, which was proposed by (Ando and Zhang, 2005).

The central idea of this method is as follows: first, define m pivot features which behave similarly in both domains, and then m predictors of the pivot features are built based on simple linear classifiers to model the correspondence between the pivot features and non-pivot features. Finally the original feature matrix $\mathbf{x}$ is projected to a low-dimensional shared feature space $\theta \cdot \mathbf{x}$. During training and test, both original features $\mathbf{x}$ and the new features $\theta \cdot \mathbf{x}$ are used.

The projection matrix θ is obtained by using the Singular Value Decomposition (SVD) on the weight matrix of the m predictors. If some non-pivot features from different domains are highly correlated to the same pivot features, then they are projected to same area in the new feature space. This is superior than Daumé III (2007) because even some context features from two domains are quite different, the *SCL* model can still estimate the correspondence between them.

There are two very interesting features in the algorithm. The first is the pivot feature predictors, which can be considered as pseudo tasks in (Ando and Zhang, 2005). The intuition is that predicting pivot features is highly related to the objective task (in the paper the task was POS tagging). Another notable feature of this algorithm is the use of SVD. SVD was employed for latent semantic association (Deerwester et al., 1990) in 1990s. It can compress the information from original feature space into a low-dimensional shared space. In *SCL* the SVD can efficiently compute θ based on the weight matrix of the m predictors. Although the *SCL* algorithm is designed for the situation where labeled data is unavailable in the target domain, the author presented a method to combine the *SCL* algorithm with labeled data in the target domain based on classifier combination techniques introduced in (Florian et al., 2004).

In sum, the key advantage of *SCL* is that it can model and utilize the correlation between features in different domains, even if these features are not the same. In addition, it doesn't need any labeled data in the target domain, which makes it useful in most practical situations.

A limitation lies in the key step of this method, i.e. defining the set of pivot features. It's unclear that how much the performance relies on the quality of the pivot feature set, and what happens if there isn't such explicit pivot features. In some cases, training data includes examples which don't have any correspondence. For instance, in the application of cross-lingual sentiment analysis, the source training data is Machine Translation results from the source language. Therefore the source domain instances include numerous translation errors, it's challenging to determine such correspondence between bad translations and the target instances.

In addition, the intuition of *SCL* is that the tasks of learning m pivot feature predictors are beneficial to the task of POS tagging, therefore jointly modeling these tasks can benefit POS tagging. Although it seems reasonable, this assumption lacks a strict justification. If we want to apply *SCL* to a new task other than POS tagging, it's difficult to tell whether the pseudo tasks are beneficial or not. For example, in the task of Named Entity Recognition(NER), many features which behave similarly in two domains may be not related to named entities, this makes the task of predicting such features somehow irrelevant to NER. Taking the following sentence in a news article as an example:

> *Since* October of *last year* , **the Hulun Buir League** promulgated a new preferential policy for opening up to the *outside world*.

"the Hulun Buir League" is an organization name. Words in italic are frequent words in news articles which have consistent senses. They can be considered as pivot features for *POS* tagging. However, they are almost irrelevant to common named entities. Therefore such pivot features are not quite beneficial to NER tagging.

Another problem is that all features are equally treated in the pivot feature predictors, thus the final shared low-dimensional space is very likely to be influenced by frequent features (Tan and Cheng, 2009). Consider again the NER application, assuming we have defined a set of important pivot features, frequent non-pivot features such as *reports* and *they* could dominate the predictors, thus low-frequency features which are important to NER will have little impact on the shared feature space. It would be beneficial if we can design an effective feature selection method to find pivot features and non-pivot features which are highly correlated to the task.

2.1.3 Generalizable Feature Selection

The assumption of usefulness of pseudo tasks in Blitzer, McDonald, and Pereira (2006)'s work needs more justification in real application. But the general idea of producing a shared feature space is an interesting direction. Jiang and Zhai (2007b) proposed a two-stage Domain Adaptation method to estimate and make use of generalizable features across domains. To distinguish this approach from other related work which also involves "two stages", we refer to it as *Generalizable Features Selection.* This method assumes there is a set of source domains $\{d_k\}_{k=1}^{K}$ with labeled training data, but no labeled data in the target domain. An important assumption is that those multiple source domains and the target domain share some features, the author called these features as *generalizable features.* This idea is similar to the use of shared feature space in (Blitzer, McDonald, and Pereira, 2006)'s work. The major difference is that the so-called *generalizable features* is a subset of original feature set $\mathbf{x}$. In contrast, (Blitzer, McDonald, and Pereira, 2006) compressed the original feature set into a low-dimensional feature space $\theta \cdot \mathbf{x}$. Another difference is that (Blitzer, McDonald, and Pereira, 2006)'s algorithm learns the new feature space using unlabeled data since it's easy to obtain "training data" for the m pseudo tasks. This framework needs multiple source domain training data to find the generalizable feature sets. Similar to (Daumé III, 2007)'s work, the drawback is that it doesn't model the correspondence between different but related features.

Roughly speaking, this framework consists of two steps:

- Domain Selection: Identify a set of generalizable features that are shared by all source domains.
- Domain Adaptation: With the generalizable feature set in hand, a classifier is trained from the union of all source domains and "labeled data" in the target domain from bootstrapping.

In the first step, the algorithm is designed to find an optimal feature selection matrix A such that $z = Ax$ is the selected generalizable features from the original feature vector x. A must satisfy two requirements: (i) each entry $a_{i,j}$ in A must be either 0 or 1; (ii) $A \times A^T = I$. With this matrix, the original weight matrix w for x in the k-th domain can be re-written as $w^k = A^T v + u^k$, where $A^T v$ corresponds to the weights for the generalizable features, and u^k is the domain-specific feature weights. The matrix A is similar to the notion of mapping matrix θ in the SCL model (Blitzer, McDonald, and Pereira, 2006).

With this definition, the key challenge is how to estimate A^* and the weights, since enumerating all possible A is computationally prohibitive. To address this problem, the author proposed two methods: (i) Jointly optimize v and A using an alternating optimization procedure, (ii) Domain Cross Validation based on a Heuristic method. Both methods can not find exact solution of the best A^*. The first approach is computational costly, since it needs to re-train classifiers iteratively. For this reason, the second approach is superior given that it only trains $2k$ classifiers if there are k source domains.

In the second stage, given a fixed A^*, the final model is trained on the combination of labeled data in the source domains and pseudo labeled data in the target domain. The weights for generalizable features are shared by all domains. Here the pseudo labeled data is gained from a bootstrapping procedure. The technique of bootstrapping is often used in semi-supervised learning. The author didn't provide any stop criteria for the bootstrapping iterations. If the bootstrapping procedure stops too early, then the pseudo labeled data has strong bias to the source domain data, only a little information about the target domain can be covered in the selected dataset. Otherwise, if the procedure stops too late, then it's likely to include many incorrect labels and out-of-domain instances. It would be beneficial if we can define a certain metric to decide when to stop the bootstrapping iterations, or train the model in the bootstrapping purely based on generalizable features.

In general this approach is similar to the idea of (Ando and Zhang, 2005). The key distinction is as follows: in this work, the assumption is that the generalizable feature space is shared by all domains, while in (Ando and Zhang, 2005) and its derivation (Blitzer, McDonald, and Pereira, 2006), the projected feature space is shared by multiple mutually beneficial tasks. Therefore the assumption in this work is more suitable for the situation where the source domain and the target domain are from a same large dataset. Another restriction of this method is that it requires more than one source domains with labeled training data. In common applications, if we don't have many source domains, a simple solution is to manually split the source domain into several subsets. However, it remains unclear what the performance is if the subsets have the same distribution. In this case it would be preferable if we can design some criteria to split the source domain dataset such that the resulting subsets can have different properties. For example, if the source domain is a large general dataset, while the target domain consists of documents about a particular topic. Then we can split the general dataset according to its topic distribution into different clusters as multiple source domains. Therefore the generalizable features learned from the source domains can encode the information which is independent from any particular topics.

2.1.4 Distributional Representation

Topic modeling is a class of generative models which recovers hidden topics from unlabeled text. It's a natural way to capture latent semantic concepts of text. Different contexts can be correlated through the latent topic distributions. In the problem of domain adaptation, latent concepts learned from multiple domains can help bridge the distribution gap across domains. For example, suppose the news articles in our source domain contains the politician name *Jon Huntsman*, but the name doesn't occur in our target domain. Instead, the name of another politician *Jingtao Hu*, who is the President of China, often occurs in the target domain. Using topic modeling we can infer that *Jon Huntsman* and *Jingtao Hu* have high distributional similarity since they have similar topic-term distributions. Furthermore, common topic modeling techniques such as *PLSA* (Hofmann, 1999) and *LDA* (Blei, Ng, and Jordan, 2003) are based on unsupervised learning. They don't require any labeled data, therefore it can be employed in many NLP tasks.

Guo et al. (2009) applied *LDA* for domain adaptation of named entity recognition. The *LDA* model is used to draw hidden topics for each token from a large set of unlabeled data containing both the source data and the target data. Then the algorithm takes topics with high conditional probability as features to train a model on the source domain, and test on the target domain. In order to predict latent topics distribution associated with each word x_i, the author proposed an interesting strategy called *virtual context document* to characterize each word x_i. Each *virtual context document* vd_{x_i} consists of context snippets in the whole corpus which contains the word x_i.

During the training of *LDA*, the vd_{x_i} for each word x_i is collected, then the *LDA* model Θ is trained on top of all virtual contexts. The authors proposed a novel method to train the *LDA*. Instead of using counts as the value of each term in (Blei, Ng, and Jordan, 2003), they used Point-Wise mutual Information between each context word and x_i. With the *LDA* model Θ on *virtual context documents*, the hidden topics for each word x_i are generated from Θ. They are finally injected to the original feature vector of each word. Such features are connections between different context words in different domains.

The latent topic distribution can be learned from any other probabilistic topic modeling methods such as *PLSA*. The main difference between *LDA* and other topic models is that *LDA* use a Dirichlet prior distribution on the topic mixture parameters corresponding to the documents. Therefore it can induce better topic distribution for each document (Blei, Ng, and Jordan, 2003).

The use of *virtual context document* is an interesting idea. However, in practice a single word can have different latent topic distribution under different surrounding contexts. Therefore it would be better if we can differentiate topic-distribution for any single word in different contexts.

Another problem of this method is that it models the topic distributions in the source and the target domains in one *LDA* model, which means the source contexts and the target contexts are generated by the same distribution. In the real-world distribution, the priori topic distribution should be diverse across domains. For example, articles from the political domain are more likely to report political news, while in the finance domain, the articles tend to cover finance-related topics.

(Xue et al., 2008) proposed an extension of *PLSA* for cross-domain text classification, which is called Topic-bridged PLSA (TPLSA). As before, they assume that the training data $\mathcal{D}_L$ in the source domain is available, while the target domain data is unlabeled $\mathcal{D}_U$. The central idea of this approach is: although the two domains are different, they still share some common latent topics. Assume the two models are sharing the same term-topic distributions, but $d_l \in \mathcal{D}_L$ is generated from the conditional distribution $Pr_(d_l|z)$, and each $d_u \in \mathcal{D}_U$ is generated from another distribution $Pr_(d_u|z)$. Based on this assumption, the author proposed to couple two independent $PLSAs$ models $Pr(d_u|z)$ and $Pr(d_l|z)$ on the source and the target domains to a joint model by a convex combination with a hyper-parameter $\lambda \in (0, 1)$, The term-topic distribution $Pr(z|w)$ is shared by the two $PLSAs$, therefore mixing topics z can bridge the two domains.

Another interesting point in this framework is as follows: in order to utilize the ground-truth classes in D_L, the author applied *must-link constraints* and *cannot-link constraint* idea from (Cohn and McCallum, 2003). For any d_l^i and d_l^j occurring in the same class, there is a must-link constraint in the log-likelihood function. Conversely, for any d_l^i and d_l^j which are not in the same class, a cannot-link constraint is injected to the log-likelihood function.

The parameters $p(z|w), p(d_l|z)$ and $p(d_u|z)$ can be estimated by an EM procedure. After that, a simple majority voting schedule is applied to determine mapping between class c and hidden topics z, and then classify each unlabeled document d_u.

One limitation of *TPLSA* is that it assumes the topic-word distribution $p(z|w)$ is constant across domains. This assumption is too strong. For example, in the document about *software*, the word *Bug* refers to software defect, but in the article about biology, it means insects, therefore the probability $p(z|bug)$ should be different in the two domains. Another problem is how to fix the hyper-parameter of λ, the setting of λ can directly determine to what extent to rely on which domain during the training. The author simply set the value of λ as 0.5, which means that two domains are equally important. But it would be better if we can determine λ automatically.

In addition, this model is specially designed for the problem of text classification, therefore, it can not be easily adapted to other NLP applications. In other tasks such as POS tagging and NER, the topic-word distribution $p(z|w)$ is more important than the document-topic distribution $p(d|z)$. The difference between $p(d_u|z)$ and $p(d_l|z)$ is less useful in such applications.

In sum, the methods of Distributional Representation recover latent topics from the both source and target domains in unsupervised fashion. The common advantage is that they don't require labeled data in the target domain, this is superior than methods such as (Daumé III, 2007) and many other methods based on priors (in Section 2.2) which also need labeled data in the target domain. The common drawback of using topic modeling is that setting the number of topics or other hyper-parameters such as λ in the *TPLSAs* is a hard problem, since we don't know the ground truth of the latent topics

2.2 Prior Based Adaptation

Priors are usually used in both discriminative and generative models. In generative models, such as Bayes models, the distribution of output y given input x is usually defined as:

$$P(y|x) = \frac{p(x|y) \cdot p(y)}{\sum\limits_{y'} p(x|y') \cdot p(y')}$$

where $p(y)$ is priori probability of output y, and $P(y|x)$ is posteriori probability. Given a test instance x, the model should find y^* that maximizes a posteriori probability (MAP) as output.

$$y^* = \arg\max_y p(y|x)$$

In the scenario of domain adaptation, the prior probabilities drawn from different domains may differ. The performance will be decreased dramatically if the model trained from the source domain is applied directly to the target domain. Therefore it's crucial to change the prior $p(y)$ estimated from the source domain to a proper value $p_T(y)$ for the target domain.

In discriminative models, priors were originally introduced as regularization term for parameter smoothing, which can influence the estimation of parameter λ. Therefore with different priors, the model distribution $p(y|x,\theta)$ learned from the same training data may differ. One commonly used prior in discriminative models is Gaussian prior $\mathcal{N}(\mu_i, diag(\sigma_i^2))$, where μ is the mean and σ is the variance. For the purpose of regularization, μ is often set as 0, then the Gaussian prior for parameter set Λ is $\sum_i \frac{\lambda_i^2}{2\sigma^2}$. Assuming the original log-likelihood of the training data under parameter setting Λ is $\mathcal{L}_\Lambda(\mathcal{D})$, by incorporating the Gaussian prior term, the log-likelihood of training data becomes:

$$\mathcal{L}_\Lambda(\mathcal{D}) - \sum_i \frac{\lambda_i^2}{2\sigma^2}$$

Using this modified log-likelihood, large parameter is penalized to avoid overfitting.

Although the notion of prior is quite different in generative models and discriminative models, we can think of the prior in both cases encodes some prior knowledge before the model estimation. In the domain adaptation, appropriate prior knowledge can help estimate the model for the target domain.

2.2.1 MAP Adaptation

Chelba and Acero (2006) designed a very simple but promising adaptation method based on Gaussian priors in log-linear model. In this work, the Maximum Entropy Markov model was applied in automatic capitalization task: given uniformly cased text, the goal is to recover the capitalization automatically. The author assumed that there is a large amount of source domain training data $\mathcal{D}_{s,l}$, and a small amount of target domain training data $\mathcal{D}_{t,l}$. At first, a common Maximum Entropy model with zero-mean Gaussian priors is trained on $\mathcal{D}_{s,l}$. This step is identical to traditional supervised learning. The feature weight matrix θ is estimated to optimize the log-likelihood of the source-domain training data.

In the next step, in order to shift the model $p(y|x;\theta)$ to match distribution of the target domain, another Maximum Entropy classifier is trained on $\mathcal{D}_{t,l}$. The most novel part in this work is: unlike common supervised learning, in this process the mean of the Gaussian prior for each feature weight is changed. If a feature weight θ_i is defined in the first model (this feature appears in $\mathcal{D}_{s,l}$, then in the new model, the mean of its Gaussian prior is set as the value in the first model. It can be viewed as "default value" for the θ_i in the second model. If the feature only occurs in the target domain, then the prior mean is set as 0 as usual.

The basic intuition of the second training process is that it keeps θ_i as close as to the value in the source-domain model, unless there is strong evidence in $\mathcal{D}_{t,l}$ to move the parameter away from the prior mean. The cost of such change is dominated by the variance of the Gaussian prior σ_i^2.

Although the authors only experimented with this approach with the Maximum Entropy Markov model, it can be easily adapted to any other discriminative models using Gaussian priors, especially for log-linear models. The advantages of this method are two-fold: (i) It's easy to implement, since it only needs to change the Gaussian prior for a given log-linear model. (ii) It doesn't rely much on labeled data in the target domain, like the title of the paper says: little data can help a lot. The reason is the "default" parameter values are obtained from the source-domain model, thus the training process on the target domain only aims to change the parameter values of the target domain-specific features. In contrast, in (Daumé III, 2007)'s work, both the general features and domain-specific features have same prior with zero-mean, and the objective of the training process is to optimize log-likelihood of both the source domain and target domain training data. Therefore the size of target training data must be large enough to estimate domain-specific parameters.

One limitation of this approach is that it's only suitable for an adaptation problem between two domains, while methods such as (Daumé III, 2007) can be applied when there are multiple source and target domains.

Although in the second training process the prior mean of different features are differentiated, the prior variance σ_i^2 is set as the same value which is optimized in the development set. It would be preferable if different features have different σ_i^2 such that the target-domain specific features and the common features have different penalty if they deviate from their prior mean.

2.2.2 Hierarchical Bayesian Framework

While (Chelba and Acero, 2006) had to train two separate models on the source and the target training data, Finkel and Manning (2009) proposed a joint model called hierarchical Bayesian domain adaptation framework. In this method, hierarchical Bayesian priors are extended to the Conditional Random Fields classifier. At first, similar to (Daumé III, 2007), each feature weight λ_i is replicated for each domain and a general version. After that, two different types of Gaussian priors are introduced in the hierarchy: the first type is domain-specific priors for each domain, and the second type is a top level domain-independent prior. The prior of each domain d controls feature weight for this domain $\lambda_{d,i}$. And the prior mean is the general version of the feature weight $\lambda_{*,i}$. $\lambda_{*,i}$ is in turn governed by the top-level domain-independent prior with zero-mean.

In such a hierarchy, the feature weights in different domains influence each other through the top level domain-independent prior. Since the domain independent parameters $\lambda_{*,i}$ are used as the prior mean of the domain-specific parameters, they can influence the value of domain-specific parameters $\lambda_{d,i}$. Conversely, the domain-specific parameters jointly affect the value of domain-independent parameters. If there is a strong evidence in the contexts of domain d to force $\lambda_{d,i}$ to be a large value, then the prior has little impact on $\lambda_{d,i}$. Conversely, If there is little evidence about $\lambda_{d,i}$, then the value is largely dominated by $\lambda_{*,i}$, which is similar to the feature weight of the source-domain model in (Chelba and Acero, 2006). The author used an interesting example to illustrate this idea: *Leeds* is normally a *Location* in many domains, but in the *sports* domain, its usually an *Organization*, since a football team is called *Leeds*.

It's shown that the method introduced in Daumé III (2007) is a special case of this method where the domain-specific prior mean is 0 and the variances $\sigma_d = \sigma_*$. The distinction between domain-specific and domain-independent prior variances makes this approach more advantageous than (Daumé III, 2007). Also it can be viewed as an extension of (Chelba and Acero, 2006)'s work, where the source domain prior and feature weights correspond to the top level prior and the general version of feature weights in this work. The key advantage of this work compared to (Chelba and Acero, 2006) is that it supports multiple domains while (Chelba and Acero, 2006) only works for two domains.

Since the key idea of this approach lies in the hierarchy of Gaussian priors, we must change the objective function (log-likelihood for statistical classifiers) of the underlying algorithm. It's not as easy as (Chelba and Acero, 2006; Daumé III, 2007) to implement. In addition, the classifiers for each domain are connected together as a big joint model. It consumes much more memory to train such a model than (Chelba and Acero, 2006).

The proposed hierarchical priors allow classifiers of different domains to share the information of the general version of the features. If the values of the feature λ_i in all domains are close to its general version, then this feature behaves similarly across domains. It corresponds to the notion of pivot feature in (Blitzer, McDonald, and Pereira, 2006)'s work. While the advantage of *SCL* algorithm is that it can model the correspondence between pivot features and non-pivot features, therefore different features in two domains can be projected to the same area in the new feature space if they are highly correlated. It could be an interesting direction if we can combine *SCL* and this work. For instance, we can learn the pivot features from the joint model, and then train the final classifier based on the new feature space $\theta \cdot \mathbf{x}$ which is provided by *SCL*.

2.2.3 Estimating Priors in Target Domain

The above two prior-based methods are designed for discriminative models with Gaussian prior. In this section we review Chan and Ng (2006)'s method, which studied estimating class priors in the target domain for the naive Bayesian model.

In this work, Naive Bayes algorithm is applied for word sense disambiguation. The author assumed that there is training data in the source domain, but no labeled data in the target domain, and further made an assumption that $p(x_k|y)$, the distribution of instance x_k given output (word sense) y, does not change across domains. Therefore estimating exact priors in the target domain is very important. Based on this assumption, they proposed a prior estimation method and the new priors are used to classify the target domain data in the Naive Bayes framework.

The overall adaptation framework can be decomposed to three steps: at first, a naive Bayes model $Model_S$ is trained from source domain data $\mathcal{D}_S$, if we apply this model to classify the target domain instances, it suffers from the problem that class priors are different across domains. Therefore in the second step, they use the posteriori probability $p_S(y|x)$ provided by $Model_S$ to perform an EM algorithm to estimate word sense prior $p_T(y)$ in the target domain. Finally, $p_T(y)$ and other parameters from $Model_S$ are used to output an adjusted posteriori probability $p_adjust(y|x)$ for the target domain data.

In this framework, the most novel and crucial part is the second step. The author proposed an EM algorithm to estimate the class priors in the target domain. Let $p_S(y|x)$ be the posteriori probability of instance x in the target domain estimated using $Model_S$. Let $p^k(y)$ and $p^k(y|x)$ be the new priori and posteriori probabilities in k-th iteration in the EM procedure. The EM algorithm first initializes $p^0(y)$ as $p_S(y)$. In the E-step in each iteration, $p^k(y|x)$ is calculated based on $\frac{p_S(y|x)}{\sum_{y'} p_S(y'|x)}$ which is weighted by $\frac{p^k(y)}{p_S(y)}$. In the M-step, $p^{k+1}(y)$ is updated by summing over $p^k(y|x)$ of all instances in the target domain. At the beginning, $p(y)$ is close to $p_S(y)$. After a number of iterations, it will be close to the true distribution of $p(y)$ in the target domain.

The author claimed that well-calibrated probabilities are important to yield an accurate estimation of $p^t(y|x_k)$, but the posteriori probabilities estimated from the Naive Bayes model doesn't meet the requirement of Calibrated Probabilities. Therefore they further proposed two methods to calibrate $p_S(y|x_k)$ instead of just using the estimation from $Model_S$. In the first method, pair-adjacent violators algorithm (PAV) (Miriam Ayer and Silverman, 1955) is deployed to modify $p_S(y|x_k)$ from $Model_S$. In the second approach, a logistic regression model trained from D_S

is used to provide posteriori probability $p_S(y|x_k)$. After estimating priors in the target domain, the new priors are applied with another parameters in $Model_S$ to estimate adjusted posteriori. The author proved that the estimation of priors gives significant improvement compared to directly applying naive Bayes model trained from D_S to D_T directly, and the Calibrated probabilities can further improve the performance dramatically.

Unlike the two prior-based methods in the previous sections, this method is designed for generative classifiers. The introduction of Calibrated Probabilities provides a critical suggestion for further research on this direction. Its main limitations are two-folds: First, since it only focuses on estimating appropriate class priors in the target domain, it doesn't address the problem where $p(x|y)$ differs across domains. Second, the algorithm in this approach is strongly coupled with the naive Bayes classifier, it would be difficult for it to be adapted to other classifiers. Despite these limitations, the general idea of estimating target domain priors for Bayes models is very interesting, since the difference of priori probabilities in two domains should be a common problem in the domain adaptation task. Furthermore, after estimating priors in the target domain, we can still use the model learned from the source domain with minor changes.

2.3 Instance Selection and Weighting

Unlike algorithms in the previous sections, which seek to explore feature space or adapt the models to accommodate the target domain, the techniques we will examine in this section mainly focus on the instance level, i.e. they are based on selecting and/or weighting instances.

In the case that $p_T(y|x)$ in the target domain differs from $p_S(y|x)$ in the source domain, if we can predict instances x_i for which $p_T(y_i|x_i)$ is quite different from $p_S(y_i|x_i)$, then we can remove these misleading instances from the source domain data. Conversely, if some instances x_i for which $p_T(y_i|x_i)$ is similar to $p_S(y_i|x_i)$, we can select these instances for inclusion into training data. In addition, to address to the problem where $p_T(x)$ is different from $p_S(x)$, one can assign high weight to the instances in the target domain to balance the empirical distribution.

2.3.1 Pseudo In-Domain Data Selection

Amittai et al. (2011) presented a data selection method called *Pseudo In-Domain Data Selection*. They used a small dataset that selected from a the large general-domain parallel corpus to train a statistical machine translation model for the target domain. This assumes that the large general domain $\mathcal{D}_S$ corpus includes some data $D_{S,T}$ which are drown from the same distribution as the target domain $\mathcal{D}_T$. Therefore the selection method is designed to dig out the $D_{S,T}$ which is hiding in $\mathcal{D}_S$. The authors refer to the selected subset as pseudo in-domain training data. The resulting system trained on a small set of pseudo in-domain data achieved equal or better performance than the model trained on the whole large data set.

The main challenge of this approach is how to evaluate the importance of the instances in the general source domain according to its relevance to the target domain. In this work, three different but similar instance selection criteria are employed for ranking and selecting instances: The first criterion is *Cross Entropy*. Assuming a target domain language model LM_t has been trained in advance, this criterion calculates cross entry $H(s, LM_t)$ between empirical distribution in the candidate segment s and the language model LM_t, and further uses it to calculate the perplexity $2^{H(s,LM_t)}$ as evaluating score. The sentences with lowest perplexity are chosen as the pseudo target domain data. The intuition of this criterion is to select instances whose distribution is similar to the target domain data.

The second criterion *Cross-Entropy Difference* calculates the difference between the cross entropy between s and the language model LM_t learned from the in-domain data, and the language model LM_s learned from the general-domain: $H(s, LM_t) - H_O(s, LM_s)$. Again, the sentences with lowest difference is selected as pseudo target-domain data. This criterion not only considers the instances with similar distribution to the target-domain data, but also biases against the instances whose distribution is close to the general domain.

The above two methods only take into account monolingual distribution. For machine translation problem, it's beneficial to jointly take into account distributions in both the source and the target languages. The last criterion *Bilingual Cross-Entropy Difference* is designed to meet this requirement. It sums *Cross-Entropy Difference* in the two languages: $[H_1(s, LM_t) - H_1(s, LM_s)] + [H_1(s, LM_t) - H_1(s, LM_s)]$.

Despite the difference of the three methods, the last step of this framework is simply to train a model on the selected training instances. The selected training dataset is much smaller than the large general source domain dataset. Experimental results showed that all three methods yield state-of-the-art performance on Machine Translation by using only a small set of pseudo target domain data. And the last approach gives best performance.

In general, this framework is very easy to implement, one just needs to build language models on the two domains and then calculate the selection metric scores. Another advantage is that it's totally independent of the underlying machine learning model and feature representation. This makes it possible to perform domain adaptation on complicated systems such as machine translation without modifying the detailed learning algorithm. Furthermore, it doesn't need any labeled data in the target domain.

A problem of this simple framework is that it only provides instance selection criteria, but doesn't define the threshold for the selection. Therefore it's hard to determine how many instances to be selected into the final training data. If we manually set a threshold to the data size or the evaluation scores, then it's likely that the resulting training dataset contains many instances whose distribution is different from the target domain, or it only includes a small set of biased instances which can not cover the distribution of the target domain.

Furthermore, it's only suitable to the scenario where the source domain contains examples similar to target domain. In practice, the source domain and the target domain are often two specific domains with limited similarity. For example, consider domain adaptation for NER task. If the source domain is news articles and the target domain is scientific papers, then such framework is not suitable because only uninformative examples can be selected from the source domain, which is not useful for the task.

Another disadvantage is that it only considers hard selection, where one example in the source domain is either discarded or kept. It would be beneficial if it allows soft selection, where each example is assigned a weight $w \in (0, 1)$. In contrast, the methods of (Xu, Xu, and Wang, 2011) and (Jiang and Zhai, 2007a) use weighing to select instances softly. The advantage of soft selection is that it's possible that the instances with low weights are useful in some cases.

2.3.2 Self-training and TrAdaBoost

Xu, Xu, and Wang (2011) studied domain adaptation algorithms for cross-lingual sentiment analysis, which addresses instance selection problem by using *TrAdaBoost* (Dai et al., 2007) and *Transfer Self-training.* In their work, the source domain dataset consists of machine translated sentiment training data from the source language, Obviously the data is very noisy but retains some useful information for the task in the target language.

The *TrAdaBoost* algorithm is an extension to the original *AdaBoost* algorithm (Freund and Schapire, 1996), which is tailored for transfer learning purposes. The central idea of applying the *TrAdaBoost* algorithm to domain adaptation is to iteratively reduce low quality source domain data, and train better model on the rest. In this framework, the union of the source-domain and the target-domain training data is used. At each iteration t, the algorithm samples the training set from the whole dataset according to a distribution $P^t(x)$, which is initialized as uniform distribution. Then a weak classifier is trained on the sampled training set. The sampling distribution $P(x)$ is updated according to the training errors. Unlike the traditional AdaBoost, classification errors made on $\mathcal{D}_{s,l}$ and $\mathcal{D}_{t,l}$ are responded in different ways, since the objective is to optimize the model for the target domain rather than the whole dataset:

- If a classification error is made on $x_i \in \mathcal{D}_{s,l}$, then x_i is likely to be useless for the adaptation, therefore the weight of $x_i \in D(x)$ in $P^t(x)$ is decreased so that x_i has less impact on the parameter estimation.
- In contrast, if a classification error is made on $x_i \in \mathcal{D}_{t,l}$, then the weight of x_i in $P^t(x)$ is increased to boost the performance on the target domain.

During this iterative learning process, low quality translations are filtered out gradually, and a better model is obtained after convergence.

The Self-training procedure employs an opposite strategy, which starts from training a classifier on the target labeled data, and then adds high-quality source domain examples iteratively. At the beginning, a classifier is trained purely using the training data in $\mathcal{D}_{t,l}$. At each iteration, the previous classifier is applied to the source domain data, and then correctly classified source domain examples with high confidence is added into the training set. After that, a new classifier is trained on the augmented training set. In this iterative procedure, these selected examples are high quality translations. The algorithm iteratively re-trains classifiers on augmented training sets, and stops when there isn't any more quality source domain instances. The author called this algorithm as *Transfer Self-training*.

The difference between the *Transfer Self-training* and the traditional self-training is that in the case of domain adaptation, the ground truth in the source domain is known so that we can determine whether an instance in $D_{s,l}$ is correctly classified or not. This is more reliable than just considering the confidence value from the classifier in the traditional self-training.

This approach is also similar to (Amittai et al., 2011)'s framework, because they both expand the training dataset according to the similarity between the source domain data and the target domain data. The key difference is that (Amittai et al., 2011)'s work doesn't require manual annotation on the target domain, the training instances are selected purely based on language model. In contrast, in the *Transfer Self-training*, instances are selected based on supervised classifier, which is more computationally costly. Another difference is that, in the *Transfer Self-training*, instances in the source domain are repeatedly added to the training set, semantic drift could happen when more instances are added. It would be better if we can design some stop criteria in which the distribution difference (for example, according to a language model) between the selected training set and the target domain is taken into account.

The common disadvantage of the proposed two solutions is that they only down-weight or discard the examples in the source domain for which $p_S(y|x)$ differs from $p_T(y|x)$, but doesn't study the difference between $p_T(x)$ and $p_S(x)$. In contrast, Jiang and Zhai (2007a)'s work, which we will review in later section, proposed a balanced bootstrapping framework to address the problem where $p_L(x)$ and $p_S(x)$ are different. Besides, the target-domain labeled instances must play a key role in both of them, but in practice it's not possible that we can have labeled data for every new domain. Finally, both algorithms train the underlying classifier repeatedly, which is computationally costly compared to methods only train one model after instance selection such as (Amittai et al., 2011).

2.3.3 Instance Weighting Framework

In the case that $p_T(x)$ is quite different from $p_S(x)$, instance weighting can be used to balance the $\tilde{p}(x)$ so that $\tilde{p}(x)$ is close to $\tilde{p}_T(x)$. Jiang and Zhai (2007a) proposed an instance weighting framework for domain adaptation, in which three different datasets are used to approximate the target domain distribution $p_T(x)$, namely the source labeled dataset $\mathcal{D}_S$, the target labeled dataset $\mathcal{D}_{T,l}$ and the target unlabeled dataset $\mathcal{D}_{T,u}$.

Assuming discriminative models are employed, the author framed the goal of adaptation as finding the optimal solution θ^* such that $p(y|x;\theta^*)$ is close to the real conditional distribution $p(y|x)$ in the target domain. Since the distribution of the target-domain data is unknown, the author proposed three complementary methods to approximate $p(y|x)$ in the target domain using $\mathcal{D}_S$, $\mathcal{D}_{T,l}$ and $\mathcal{D}_{T,u}$ respectively.

The first method only relies on D_S. The idea is to use $p(y|x;\theta)$ that learned from D_S to approximate $p_t(y|x)$. This is similar to the naive adaptation method in which the model learned from the source domain is directly applied to the target domain instances. The difference is that an instance weighting term $\frac{p_t(x^s)}{p_s(x^s)}$ is added to the objective function to measure the distribution difference between $p_t(x^s)$ and $p_s(x^s)$. So the challenge becomes how to evaluate $\frac{p_t(x^s)}{p_s(x^s)}$ for each instance in D_S.

The second method estimates $p(y|x;\theta)$ from $D_{T,l}$, this is identical to the standard supervised learning, since the amount of $\mathcal{D}_{t,l}$ is small, this method can not yield good performance.

In the last method, if we know $p_t(y|x_i;\theta)$ for each $x_i \in D_{T,u}$ and each label $y \in \mathcal{Y}$, then we can weight each possible pair of (x_i, y), and then add the $D_{T,u}$ to the training data. The problem here is how to estimate $p_t(y|x_i;\theta)$.

Given the above three complementary solutions, the author then proposed a general weighting framework which combines them together as a joint model. Four types of weighting parameters are crucial to this framework, namely λ, α, β and γ. For each training example $(x_i^s, y_i^s \in \mathcal{D}_S)$, the value of α_i indicates how likely $p_t(y_i^s|x_i^s)$ is close to $p_s(y_i^s|x_i^s)$. Large α_i means two distributions are close while small α_i means they are pretty different such that $p_t(y_i^s|x_i^s)$ can not be trusted to learn a classifier for the target domain. To estimate the value of α_i, the authors applied a simple strategy similar to self-learning: at first, a model $\theta^{t,l}$ is trained on $\mathcal{D}_{t,l}$ to approximate the distribution $p_t(y|x)$, then each instance x_i^s is classified by the model $\theta^{t,l}$. If it's correctly classified, $p_t(y_i^s|x_i^s)$ is pretty close to $p_s(y_i^s|x_i^s)$. Conversely, if it's misclassified, then $p_t(y_i^s|x_i^s)$ differs much from $p_s(y_i^s|x_i^s)$. Therefore, for instances in $\mathcal{D}_s$ which are misclassified with highest confidence, the value of α_i is set to 0, α_i of all others are set to 1. This is closely related to the idea of *Transfer Self-training* proposed by (Xu, Xu, and Wang, 2011), the difference is (Xu, Xu, and Wang, 2011)'s algorithm is an iterative process, it adds source domain labeled instances repeatedly.

For each $x_k^{t,u} \in \mathcal{D}_{t,u}$, and each possible label $y \in \mathcal{Y}$, parameter $\gamma_k(y)$ indicates how likely y is a correct label for $x_k^{t,u}$. The estimating $\gamma_k(y)$ is related to the weighting of pseudo labeled data in semi-supervised learning. This paper discussed two types of solutions: (i) set $\gamma_k(y)$ as a function of model parameter θ such as $\gamma_k(y) = p(y|x^{t,u};\theta)$. (ii) set $\gamma_k(y)$ in a bootstrapping manner, in which the value of γ is updated iteratively.

Finally the three parameters $\lambda_s, \lambda_{t,l}$ and $\lambda_{t,u}$ combine the three approximation methods together in the learning objective function. For the domain adaptation, more emphasis should be paid to the target domain, so $\lambda_{t,l}$ and $\lambda_{t,u}$ should be larger than λ_s. Similar to this work, (Jiang and Zhai, 2007b) also emphasizes the target domain instance in the training objective function. The difference is that in (Jiang and Zhai, 2007b)'s work this is done by setting a lower penalty on the feature weights in the target domain training instances.

The beauty of this framework is that it doesn't only down-weight the instances for which $p(y|x)$ is different in two domains, but also addresses the problem where $p(x)$ differs in different domains by using the instances in the target domain. In the traditional bootstrapping, instances in $D_{t,u}$ are selected to the training set based on their predication confidence. After this step, the selected instances are considered as important as the data in $D_{s,l}$. In contrast, this work suggests that the performance may be influenced by the difference between $p_S(x)$ and $p_T(x)$. Therefore the instances from $\mathcal{D}_{t,u}$ are attached high weights such that the empirical distribution $\tilde{p}(x)$ is balanced towards $p_T(x)$.

Another significant advantage lies in the flexibility of this framework, it takes advantage of both labeled and unlabeled data in the target domain. If labeled data in the target domain is unavailable, the algorithm would still work. It can be shown that the standard bootstrapping is a special case of this framework.

One problem of this framework is that there are several hyper-parameters to tune, and it remains unclear how to set the hyper-parameter β, which is used to approximate $\frac{p_t(x_i^s)}{p_s(x_i^s)}$. Compared to the pseudo in-domain data selection method in (Amittai et al., 2011)'work, this framework is much more complex. It requires several training process in order to set the parameters, which is computational costly.

Although this framework can be applied to various machine learning models (in the paper they used logistic regression), we still need to modify the log-likelihood function in the original training process in order to accommodate these weighting parameters. Finally, for the models that globally model structured data, each instance stands for a set of instances in a structure, and the output is a set of assignments. Consider the CRFs model for POS tagging, each instance x is a sequence of words, and the output y is the sequence of labels for the sentence. It's too coarse to perform the instance selection on sentence level. For instance, if some words of a sentence have desired distribution but others do not, it's not easy to decide whether to select or discard the whole sentence. On the other hand, if we use models that locally model each word such as logistic regression, it suffers from the problem that the predications are not globally optimized. It would be preferable if we can enhance this framework such that structured learning algorithms can be applied.

2.4 Conclusions

Domain adaptation approaches in NLP tasks can be roughly categorized into three classes: *Feature Space Transformation*, *Prior Adaptation* and *Instance Selection and Weighting*. Following this categorization, we reviewed some representative work in each category. In general the algorithms we studied in this section can be applied to various machine learning models and NLP tasks. In the rest of this section, we summarize these algorithms in several aspects: Supervised or unsupervised, dependency on the underlying models, assumption of data availability, and computational cost. Table 3 shows a brief comparison according to these aspects.

Supervised vs. unsupervised In some algorithms, the central part of the adaptation is conducted in unsupervised fashion (Blitzer, McDonald, and Pereira, 2006; Guo et al., 2009; Amittai et al., 2011). It doesn't rely on the labels in the corpora. For example, Amittai et al. (2011) addressed the instance selection problem based on language model distributions. Guo et al. (2009) learned latent semantic association using *LDA*. In Blitzer, McDonald, and Pereira (2006)'s work, although the pivot feature predictors are learned in a supervised way, the training data of the predictors can be obtained automatically from unlabeled documents. Other algorithms perform the adaptation using supervised approaches. For instance, Jiang and Zhai (2007b) extracted generalizable features using the labeled data in the source domains. Chelba and Acero (2006) estimated domain-specific feature weights under the supervision of labeled data in the target domain. Xu, Xu, and Wang (2011)'s algorithm weights instances in the source domain under

Algorithm	Supervised / unsupervised	Dependency on the underlying models	Assumption of data availability	Computational cost
Daumé III (2007)	Unsupervised	Weak	Training data in the target domain	Low
Blitzer, McDonald, and Pereira (2006)	Unsupervised	Weak	-	Low
Jiang and Zhai (2007b)	Supervised	Moderate	Multiple source domains	High
Guo et al. (2009)	Unsupervised	Weak	-	Low
Xue et al. (2008)	Semi-supervised	Strong	-	Low
Chelba and Acero (2006)	Supervised	Moderate	Training data in the target domain	Low
Finkel and Manning (2009)	Supervised	Moderate	Training data in the target domain	Moderate
Chan and Ng (2006)	Unsupervised	Strong	-	Low
Amittai et al. (2011)	Unsupervised	Very weak	-	Low
Xu, Xu, and Wang (2011)	Supervised	Weak	Training data in the target domain	High
Jiang and Zhai (2007a)	Supervised	Weak	-	High

Table 3: Summarization of the algorithms. ”-” means there isn’t any special restriction

the supervision of labeled in the target domain.

Dependency on underlying models Most algorithms in *Instance Selection and Weighting* are agnostic of the underlying machine learning algorithms. Therefore they are preferable for the cases where the underlying tasks or classifiers are complex, such as statistical machine translation. For example, Amittai et al. (2011)'s work can be easily applied to various applications since it can be considered as a preprocessing step before conducting any applications. However, some of them are not suitable for a structured learning algorithms because they are working on the instance level. For instance, in Jiang and Zhai (2007a)'s framework, if the basic classifier is structured learning model such as CRFs, the instance becomes sentences rather than words or chunks. It would be too coarse to do weighting on the sentence level. Since *Feature Space Transformation* algorithms aim to provide predicative feature representation, in theory they are applicable to most NLP tasks which are based on feature vectors such as POS tagging and named entity recognition. Algorithms in *Prior Adaptation* are highly coupled with the basic learning models. Therefore we need to modify the original learning objective functions for this type of methods. The algorithms in Chelba and Acero (2006) and Finkel and Manning (2009) are suitable for discriminative classifiers with Gaussian priors, while the idea in Chan and Ng (2006) is applicable to Bayes models which depend on priori distributions.

Assumption of data availability The assumption of data availability is critical to domain adaptation. (Daumé III, 2007; Chelba and Acero, 2006; Finkel and Manning, 2009; Xu, Xu, and Wang, 2011) require available training data in the target domain. Among them, (Daumé III, 2007; Finkel and Manning, 2009) can be viewed as multi-domain learning methods, in which the same task on different domains mutually benefit each other. In contrast, (Chelba and Acero, 2006; Xu, Xu, and Wang, 2011) are designed for the cases where only a small amount of training data in the target domain is available. (Blitzer, McDonald, and Pereira, 2006; Jiang and Zhai, 2007b; Guo et al., 2009; Xue et al., 2008; Chan and Ng, 2006; Amittai et al., 2011) don't need labeled data in the target domain. Therefore they are preferable in most practical settings. Jiang and Zhai (2007a)'s framework is very flexible in the sense that it is independent of the availability of labeled data in the target domain.

Computational cost Some algorithms need to train the underlying machine learning models for several times, either because the frameworks need to iteratively select features or instances, or because they need to estimate domain-specific feature weights. (Jiang and Zhai, 2007b)'s framework iteratively trains models in order to select the generalizable features. In addition, a bootstrapping procedure is required in the final model to provide training data in the target domain. (Jiang and Zhai, 2007a; Xu, Xu, and Wang, 2011) needs to iteratively train models to re-weight instances in $\mathcal{D}_s$ and $\mathcal{D}_{t,u}$. Therefore their algorithms are more computational costly than others during the training stage. (Chelba and Acero, 2006)'s framework only needs to train two classifiers, where the second model is to adjust target-domain specific feature weights. It's less computational expensive compared to the above two methods.

3 Applications

In this section, we summarize some common NLP applications of domain adaptation in recent publications. The adaptation algorithms are reviewed in the previous section, therefore in this section we will focus on different applications and their experimental settings.

Although some algorithms we studied in Section 2 are applicable to general machine learning problems, most of them are designed for special applications. Also since there isn't a standard setting for domain adaptation, various experimental setups are studied in the previous work. In Table 3 we summarize the properties of different adaptation applications, and list common settings and suitable algorithms for each application. In the remaining part of this section, we compare different experimental settings in detail.

Adaptation between different corpora

This is the most common setting of domain adaptation in NLP applications. Finkel and Manning (2009) experimented adaptation between CoNLL data and MUC data. The main domain difference is that the first dataset is in British English while the latter is in American English. Arnold, Nallapati, and Cohen (2008) used MUC data and CSPACE personal email data as two separate domains for tagging person names. The gap between MUC and CSPACE is much larger than that between CoNLL and MUC. Chelba and Acero (2006) studied domain adaptation for automatic capitalization from Wall Street Journal dataset to Broadcast News dataset.

The key problem of this setting is that the task definitions differ in different datasets. Taking the NER task as an example, the definitions of entity boundaries and types differ across domains. Therefore the task in the target domain is limited

Setting	Application	Challenge	Suitable algorithm
Different corpora	Named Entity Recognition	(1)Same entity are represented by different words, (2) Contexts/Vocabulary difference	Finkel and Manning (2009) Arnold, Nallapati, and Cohen (2008)
	POS Tagging	(1) Homonym (2) Contexts/Vocabulary difference	Daumé III (2007) Jiang and Zhai (2007b) Blitzer, McDonald, and Pereira (2006)
	Automatic Capitalization	Contexts and Vocabulary differences	Chelba and Acero (2006) Daumé III (2007)
	Word Sense Disambiguation	Sense priori distribution difference	Chan and Ng (2006)
General to specific	Machine Translation	(1) Misleading instances in other domains (2) Model Complexity	Amittai et al. (2011)
Sub-topics of the same corpus	Named Entity Recognition	(1)Same entity are represented by different words (2) Contexts/Vocabulary difference	Daumé III (2007) Wu et al. (2009) Jiang and Zhai (2007b)
	Sentiment Analysis	Different aspects of opinions	Blitzer, Dredze, and Pereira (2007) Blitzer, Dredze, and Pereira (2007)
Cross-lingual	Sentiment Analysis	Translation Quality	Xu, Xu, and Wang (2011)

Table 4: Application Summarization

to the overlapped entity types. For instance, Arnold, Nallapati, and Cohen (2008) restricted the task as finding person name mentions when performing adaptation between MUC and CSPACE.

Adaptation between topics of the same corpus

Some corpora can be divided by genres and topics. For instance: given that genes and proteins associated with different organisms have various spellings and occur in quite different contexts as well, Jiang and Zhai (2007b) split BioCreAtIvE dataset into 3 domains according to the organisms, namely *fly*, *mouse* and *yeast*. Daumé III (2007) did experiments with the 6 subsets of ACE 2005 corpus, namely *bc*, *bn*, *cts*, *nw*, *wl* and *un*. Wu et al. (2009) manually classified the documents in ACE 2005 corpus into 4 topics: military operations, political relationship or politicians, terrorism-related, and all others. Blitzer, Dredze, and Pereira (2007) and Glorot, Bordes, and Bengio (2011) considered customer reviews of different products in a large sentiment analysis corpus as different domains.

An advantage of the adaptation between subsets of the same corpus is that the task definition is identical. For example, in the NER task, the definition of entity types and boundaries is identical, therefore the experiments can be done in all types of names. In addition, the split according to genres and/or topics is natural in practical scenarios. The difference and similarity between domains are more clear compared to the first type of setup. For example, in the sentiment analysis task, reviews about different tasks are based on different aspects. For instance, in the reviews of *Kitchen applications*, the words such as *malfunctioning* or *reliable* are about subjective opinions. While in the domain of *DVD*, reviews may contain words like *horrific* or *thrilling* (Glorot, Bordes, and Bengio, 2011).

A potential problem of this setup is that the data size of both training and test data is probably not big enough to obtain statistically significant results. For example, in the ACE 2005 corpus, there are merely 39 documents in *cts*, *49* documents in *un*, and *60* documents in *bc*. This is too small compared to the common benchmark for named entity recognition experiments. For example in the CoNLL-2003 NER shared task, the English training and test data consists of 946 and 231 articles respectively. This drawback makes it difficult to show statistically significant improvement.

Cross-lingual adaptation

Domain adaptation task in this category assumes that there is enough training data in one language, and the goal is to build a classifier for the dataset in another language. (Xu, Xu, and Wang, 2011) studied this setting for sentiment analysis, i.e. transfer the sentiment analysis model learned from the source language to the target language. They performed experiments between English and Chinese for two sub-tasks: sentence-level opinionated sentence recognition, using dataset of NTCIR-7 Multilingual Opinion Analysis Tasks (Seki et al., 2008), another is document-level review polarity classification using datasets in (Wan, 2009). The English sentences are translated to Chinese by Google Translate as the source domain training data. Different from other domain adaptation settings, the domain difference in this scenario is mainly caused by Machine Translation errors. Therefore the empirical results are to some extent determined by the Machine Translation system.

Adaptation from general domain to specific domain

In the case that a large general-domain training data is available for a task, if we want to build model for a particular domain, it's preferable to use some labeled instances which are similar to instances in the target domain, instead of using the whole general-domain corpus. Because the target domain should have its own distributions, instances in the general-domain corpus which are in different domains will harm the performance. Therefore some domain adaptation algorithms are designed for this setting to improve the performance for the target domain. Amittai et al. (2011) proposed a solution for this setting based on instance selection. In their work, the target domain data is the corpus from the International Workshop on Spoken Language Translation (IWSLT) Chinese-to-English DIALOG task, while the general source domain is a combination of various datasets. Compared to other settings, the key assumption in this setting is that the instances from the target domain are hidden in the large general corpus. Therefore instance selection methods such as Amittai et al. (2011) is suitable to this setting.

References

Amittai, Axelrod, Xiaodong He, Jianfeng Gao, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, July.

Ando, Rie Kubota and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6.

Arnold, Andrew, Ramesh Nallapati, and William W. Cohen. 2008. Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL*, pages 245–253.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Blitzer, John, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

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

Bunescu, Razvan C. and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*.

Chan, Yee Seng and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *ACL*.

Chelba, Ciprian and Alex Acero. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*.

Cohn, D., Caruana R. and McCallum. 2003. A. semi-supervised clustering with user feedback.technical report tr2003-1892, cornell university.

Cucerzan, Silviu. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716.

Dai, Wenyuan, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for transfer learning. In *ICML*, pages 193–200.

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

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

Finkel, Jenny Rose and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *HLT-NAACL*, pages 602–610.

Florian, Radu, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *HLT-NAACL*.

Freund, Yoav and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *ICML*, pages 148–156.

Gabrilovich, Evgeniy and Shaul Markovitch. 2005. Feature generation for text categorization using world knowledge. In *IJCAI*, pages 1048–1053.

Gabrilovich, Evgeniy and Shaul Markovitch. 2006. Enhancing text categorization with encyclopedic knowledge. In *AAAI*, pages 1301–1306.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

Guo, Honglei, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. 2009. Domain adaptation with latent semantic association for named entity recognition. In *HLT-NAACL*, pages 281–289.

Hofmann, Thomas. 1999. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57.

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

Jiang, Jing and ChengXiang Zhai. 2007b. A two-stage approach to domain adaptation for statistical classifiers. In *CIKM*, pages 401–410.

Liu, Chen, Sai Wu, Shouxu Jiang, and Anthony Tung. 2012. Cross domain search by exploiting wikipedia. In *ICDE*.

Mihalcea, Rada and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242.

Miriam Ayer, H. D. Brunk, G. M. Ewing W. T. Reid and Edward Silverman. 1955. An empirical distribution function for sampling with incomplete information. In *Annals of Mathematical Statistics*.

Pan, Sinno Jialin and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Ratinov, Lev-Arie, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*, pages 1375–1384.

Seki, Yohei, David Kirk Evans, Lun-Wei Ku, Le Sun, Hsin-Hsi Chen, and Noriko Kando. 2008. Overview of multilingual opinion analysis task at ntcir-7.

Tan, Songbo and Xueqi Cheng. 2009. Improving scl model for sentiment-transfer learning. In *HLT-NAACL (Short Papers)*, pages 181–184.

Wan, Xiaojun. 2009. Co-training for cross-lingual sentiment classification. In *47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*.

Wu, Dan, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In *EMNLP*, pages 1523–1532.

Xu, Ruifeng, Jun Xu, and Xiaolong Wang. 2011. Instance level transfer learning for cross lingual opinion analysis. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, Portland, Oregon, June.

Xue, Gui-Rong, Wenyuan Dai, Qiang Yang, and Yong Yu. 2008. Topic-bridged plsa for cross-domain text classification. In *SIGIR*, pages 627–634.