7-2015

A Hassle-free Unsupervised Domain Adaptation Method using Instance Similarity Features

Jianfei YU
Singapore Management University, jfyu.2014@phdis.smu.edu.sg

Jing JIANG
Singapore Management University, jingjiang@smu.edu.sg

Citation

YU, Jianfei and Jing JIANG. A Hassle-free Unsupervised Domain Adaptation Method using Instance Similarity Features. (2015). *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Beijing, China, July 26-31, 2015*. 2, 168-173.
Available at: https://ink.library.smu.edu.sg/sis_research/2981

A Hassle-Free Unsupervised Domain Adaptation Method Using Instance Similarity Features

Jianfei Yu
School of Information Systems
Singapore Management University
jfyu.2014@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

Abstract

We present a simple yet effective unsupervised domain adaptation method that can be generally applied for different NLP tasks. Our method uses unlabeled target domain instances to induce a set of instance similarity features. These features are then combined with the original features to represent labeled source domain instances. Using three NLP tasks, we show that our method consistently outperforms a few baselines, including SCL, an existing general unsupervised domain adaptation method widely used in NLP. More importantly, our method is very easy to implement and incurs much less computational cost than SCL.

1 Introduction

Domain adaptation aims to use labeled data from a source domain to help build a system for a target domain, possibly with a small amount of labeled data from the target domain. The problem arises when the target domain has a different data distribution from the source domain, which is often the case. In NLP, domain adaptation has been well studied in recent years. Existing work has proposed both techniques designed for specific NLP tasks (Chan and Ng, 2007; Daume III and Jagarlamudi, 2011; Yang et al., 2012; Plank and Moschitti, 2013; Hu et al., 2014; Nguyen et al., 2014; Nguyen and Grishman, 2014) and general approaches applicable to different tasks (Blitzer et al., 2006; Daumé III, 2007; Jiang and Zhai, 2007; Dredze and Crammer, 2008; Titov, 2011). With the recent trend of applying deep learning in NLP, deep learning-based domain adaptation methods (Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2014) have also been adopted for NLP tasks (Yang and Eisenstein, 2015).

There are generally two settings of domain adaptation. We use *supervised domain adaptation* to refer to the setting when a small amount of labeled target data is available, and when no such data is available during training we call it *unsupervised domain adaptation*.

Although many domain adaptation methods have been proposed, for practitioners who wish to avoid implementing or tuning sophisticated or computationally expensive methods due to either lack of enough machine learning background or limited resources, simple approaches are often more attractive. A notable example is the frustratingly easy domain adaptation method proposed by Daumé III (2007), which simply augments the feature space by duplicating features in a clever way. However, this method is only suitable for supervised domain adaptation. A later semi-supervised version of this easy adaptation method uses unlabeled data from the target domain (Daumé III et al., 2010), but it still requires some labeled data from the target domain. In this paper, we propose a general unsupervised domain adaptation method that is almost equally hassle-free but does not use any labeled target data.

Our method uses a set of unlabeled target instances to induce a new feature space, which is then combined with the original feature space. We explain analytically why the new feature space may help domain adaptation. Using a few different NLP tasks, we then empirically show that our method can indeed learn a better classifier for the target domain than a few baselines. In particular, our method performs consistently better than or competitively with Structural Correspondence Learning (SCL) (Blitzer et al., 2006), a well-known unsupervised domain adaptation method in NLP. Furthermore, compared with SCL and other advanced methods such as the marginalized structured dropout method (Yang and Eisenstein, 2014) and a recent feature embedding method (Yang and

*Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics
and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 168–173,
Beijing, China, July 26-31, 2015.

Eisenstein, 2015), our method is much easier to implement.

In summary, our main contribution is a simple, effective and theoretically justifiable unsupervised domain adaptation method for NLP problems.

2 Adaptation with Similarity Features

We first introduce the necessary notation needed for presenting our method. Without loss of generality, we assume a binary classification problem where each input is represented as a feature vector $\boldsymbol{x}$ from an input vector space $\mathcal{X}$ and the output is a label $y \in \{0, 1\}$. This assumption is general because many NLP tasks such as text categorization, NER and relation extraction can be cast into classification problems and our discussion below can be easily extended to multi-class settings. We further assume that we have a set of labeled instances from a source domain, denoted by $D^{\mathrm{s}} = \{(\boldsymbol{x}_i^{\mathrm{s}}, y_i^{\mathrm{s}})\}_{i=1}^N$. We also have a set of unlabeled instances from a target domain, denoted by $D^{\mathrm{t}} = \{\boldsymbol{x}_j^{\mathrm{t}}\}_{j=1}^M$. We assume a general setting of learning a linear classifier, which is essentially a weight vector $\boldsymbol{w}$ such that $\boldsymbol{x}$ is labeled as 1 if $\boldsymbol{w}^\top \boldsymbol{x} \geq 0$.[1]

A naive method is to simply learn a classifier from D^{s}. The goal of unsupervised domain adaptation is to make use of both D^{s} and D^{t} to learn a good $\boldsymbol{w}$ for the target domain. It has to be assumed that the source and the target domains are similar enough such that adaptation is possible.

2.1 The Method

Our method works as follows. We first randomly select a subset of target instances from D^{t} and normalize them. We refer to the resulting vectors as *exemplar vectors*, denoted by $\mathcal{E} = \{\boldsymbol{e}^{(k)}\}_{k=1}^K$. Next, we transform each source instance $\boldsymbol{x}$ into a new feature vector by computing its similarity with each $\boldsymbol{e}^{(k)}$, as defined below:

$$g(\boldsymbol{x}) = [s(\boldsymbol{x}, \boldsymbol{e}^{(1)}), \ldots, s(\boldsymbol{x}, \boldsymbol{e}^{(K)})]^\top, \quad (1)$$

where $\top$ indicates transpose and $s(\boldsymbol{x}, \boldsymbol{x}')$ is a similarity function between $\boldsymbol{x}$ and $\boldsymbol{x}'$. In our work we use dot product as s.[2] Once each labeled source domain instance is transformed into a K-dimensional vector by Equation 1, we can append this vector to the original feature vector of the source instance and use the combined feature vectors of all labeled source instances to train a classifier. To apply this classifier to the target domain, each target instance also needs to add this K-dimensional induced feature vector.

It is worth noting that the exemplar vectors are randomly chosen from the available target instances and no special trick is needed. Overall, the method is fairly easy to implement, and yet as we will see in Section 3, it performs surprisingly well. We also want to point out that our instance similarity features bear strong similarity to what was proposed by Sun and Lam (2013), but their work addresses a completely different problem and we developed our method independently of their work.

2.2 Justification

In this section, we provide some intuitive justification for our method without any theoretical proof.

Learning in the Target Subspace

Blitzer et al. (2011) pointed out that the hope of unsupervised domain adaptation is to "couple" the learning of weights for target-specific features with that of common features. We show our induced feature representation is exactly doing this.

First, we review the claim by Blitzer et al. (2011). We note that although the input vector space $\mathcal{X}$ is typically high-dimensional for NLP tasks, the actual space where input vectors lie can have a lower dimension because of the strong feature dependence we observe with NLP tasks. For example, binary features defined from the same feature template such as the previous word are mutually exclusive. Furthermore, the actual low-dimensional spaces for the source and the target domains are usually different because of domain-specific features and distributional difference between the domains. Borrowing the notation used by Blitzer et al. (2011), define subspace $\mathcal{X}_{\mathrm{s}}$ to be the (lowest dimensional) subspace of $\mathcal{X}$ spanned by all source domain input vectors. Similarly, a subspace $\mathcal{X}_{\mathrm{t}}$ can be defined. Define $\mathcal{X}_{\mathrm{s,t}} = \mathcal{X}_{\mathrm{s}} \bigcap \mathcal{X}_{\mathrm{t}}$, the shared subspace between the two domains. Define $\mathcal{X}_{\mathrm{s},\perp}$ to be the subspace that is orthogonal to $\mathcal{X}_{\mathrm{s,t}}$ but together with $\mathcal{X}_{\mathrm{s,t}}$ spans $\mathcal{X}_{\mathrm{s}}$, that is, $\mathcal{X}_{\mathrm{s},\perp} + \mathcal{X}_{\mathrm{s,t}} = \mathcal{X}_{\mathrm{s}}$. Similarly we can define $\mathcal{X}_{\perp,\mathrm{t}}$. Essentially $\mathcal{X}_{\mathrm{s,t}}$, $\mathcal{X}_{\mathrm{s},\perp}$ and $\mathcal{X}_{\perp,\mathrm{t}}$ are the shared

[1] A bias feature that is always set to be 1 can be added to allow a non-zero threshold.

[2] We find that normalizing the exemplar vectors results in better performance empirically. On the other hand, if we normalize both the exemplar vectors and each instance $\boldsymbol{x}$, i.e. if we use cosine similarity as s, the performance is similar to not normalizing $\boldsymbol{x}$.

subspace and the domain-specific subspaces, and they are mutually orthogonal.

We can project any input vector $\boldsymbol{x}$ into the three subspaces defined above as follows:

$$\boldsymbol{x} = \boldsymbol{x}_{\mathrm{s,t}} + \boldsymbol{x}_{\mathrm{s},\perp} + \boldsymbol{x}_{\perp,\mathrm{t}}.$$

Similarly, any linear classifier $\boldsymbol{w}$ can be decomposed into $\boldsymbol{w}_{\mathrm{s,t}}$, $\boldsymbol{w}_{\mathrm{s},\perp}$ and $\boldsymbol{w}_{\perp,\mathrm{t}}$, and

$$\boldsymbol{w}^\top \boldsymbol{x} = \boldsymbol{w}_{\mathrm{s,t}}^\top \boldsymbol{x}_{\mathrm{s,t}} + \boldsymbol{w}_{\mathrm{s},\perp}^\top \boldsymbol{x}_{\mathrm{s},\perp} + \boldsymbol{w}_{\perp,\mathrm{t}}^\top \boldsymbol{x}_{\perp,\mathrm{t}}.$$

For a naive method that simply learns $\boldsymbol{w}$ from D^{s}, the learned component $\boldsymbol{w}_{\perp,\mathrm{t}}$ will be $\mathbf{0}$, because the component $\boldsymbol{x}_{\perp,\mathrm{t}}$ of any source instance is $\mathbf{0}$, and therefore the training error would not be reduced by any non-zero $\boldsymbol{w}_{\perp,\mathrm{t}}$. Moreover, any non-zero $\boldsymbol{w}_{\mathrm{s},\perp}$ learned from D^{s} would not be useful for the target domain because for all target instances we have $\boldsymbol{x}_{\mathrm{s},\perp} = \mathbf{0}$. So for a $\boldsymbol{w}$ learned from D^{s}, only its component $\boldsymbol{w}_{\mathrm{s,t}}$ is useful for domain transfer.

Blitzer et al. (2011) argues that with unlabeled target instances, we can hope to "couple" the learning of $\boldsymbol{w}_{\perp,\mathrm{t}}$ with that of $\boldsymbol{w}_{\mathrm{s,t}}$. We show that if we use only our induced feature representation without appending it to the original feature vector, we can achieve this. We first define a matrix $M_{\mathcal{E}}$ whose column vectors are the exemplar vectors from $\mathcal{E}$. Then $g(\boldsymbol{x})$ can be rewritten as $M_{\mathcal{E}}^\top \boldsymbol{x}$. Let $\boldsymbol{w}'$ denote a linear classifier learned from the transformed labeled data. $\boldsymbol{w}'$ makes prediction based on $\boldsymbol{w}'^\top M_{\mathcal{E}}^\top \boldsymbol{x}$, which is the same as $(M_{\mathcal{E}}\boldsymbol{w}')^\top \boldsymbol{x}$. This shows that the learned classifier $\boldsymbol{w}'$ for the induced features is equivalent to a linear classifier $\overline{\boldsymbol{w}} = M_{\mathcal{E}}\boldsymbol{w}'$ for the original features.

It is not hard to see that $M_{\mathcal{E}}\boldsymbol{w}'$ is essentially $\sum_k w'_k \boldsymbol{e}^{(k)}$, i.e. a linear combination of vectors in $\mathcal{E}$. Because $\boldsymbol{e}^{(k)}$ comes from $\mathcal{X}_{\mathrm{t}}$, we can write $\boldsymbol{e}^{(k)} = \boldsymbol{e}^{(k)}_{\mathrm{s,t}} + \boldsymbol{e}^{(k)}_{\perp,\mathrm{t}}$. Therefore we have

$$\overline{\boldsymbol{w}} = \underbrace{\sum_k w'_k \boldsymbol{e}^{(k)}_{\mathrm{s,t}}}_{\overline{\boldsymbol{w}}_{\mathrm{s,t}}} + \underbrace{\sum_k w'_k \boldsymbol{e}^{(k)}_{\perp,\mathrm{t}}}_{\overline{\boldsymbol{w}}_{\perp,\mathrm{t}}}.$$

There are two things to note from the formula above. (1) The learned classifier $\overline{\boldsymbol{w}}$ does not have any component in the subspace $\mathcal{X}_{\mathrm{s},\perp}$, which is good because such a component would not be useful for the target domain. (2) The learned $\overline{\boldsymbol{w}}_{\perp,\mathrm{t}}$ will unlikely be zero because its learning is "coupled" with the learning of $\overline{\boldsymbol{w}}_{\mathrm{s,t}}$ through $\boldsymbol{w}'$. In effect, we pick up target specific features that correlate with useful common features.

In practice, however, we need to append the induced features to the original features to achieve good adaptation results. One may find this counter-intuitive because this results in an expanded instead of restricted hypothesis space. Our explanation is that because of the typical L_2 regularizer used during training, there is an incentive to shift the weight mass to the additional induced features. The need to combine the induced features with original features was also reported in previous domain adaptation work such as SCL (Blitzer et al., 2006) and marginalized denoising autoencoders (Chen et al., 2012).

Reduction of Domain Divergence

Another theory on domain adaptation developed by Ben-David et al. (2010) essentially states that we should use a hypothesis space that can achieve low error on the source domain while at the same time making it hard to separate source and target instances. If we use only our induced features, then $\mathcal{X}_{\mathrm{s},\perp}$ is excluded from the hypothesis space. This is likely to make it harder to distinguish source and target instances. To verify this, in Table 1 we show the following errors based on three feature representations: (1) The training error on the source domain ($\hat{\varepsilon}_{\mathrm{s}}$). (2) The classification error when we train a classifier to separate source and target instances. (3) The error on the target domain using the classifier trained from the source domain ($\hat{\varepsilon}_{\mathrm{t}}$). ISF- means only our induced instance similarity features are used while ISF uses combined feature vectors. The results show that ISF achieves relatively low $\hat{\varepsilon}_{\mathrm{s}}$ and increases the domain separation error. These two factors lead to a reduction in $\hat{\varepsilon}_{\mathrm{t}}$.

features	$\hat{\varepsilon}_{\mathrm{s}}$	domain separation error	$\hat{\varepsilon}_{\mathrm{t}}$
Original	0.000	0.011	0.283
ISF-	0.120	0.129	0.315
ISF	0.006	0.062	**0.254**

Table 1: Three errors of different feature representations on a spam filtering task. K is 200 for ISF- and ISF. We expect a low $\hat{\varepsilon}_{\mathrm{t}}$ when $\hat{\varepsilon}_{\mathrm{s}}$ is low and domain separation error is high.

Difference from EA++

The easy domain adaptation method EA proposed by Daumé III (2007) has later been extended to a semi-supervised version EA++ (Daumé III et al., 2010), where unlabeled data from the target domain is also used. Theoretical justifications for both EA and EA++ are given by Kumar et al.

(2010). Here we briefly discuss how our method is different from EA++ in terms of using unlabeled data. In both EA and EA++, since labeled target data is available, the algorithms still learn two classifiers, one for each domain. In our algorithm, we only learn a single classifier using labeled data from the source domain. In EA++, unlabeled target data is used to construct a regularizer that brings the two classifiers of the two domains closer. Specifically, the regularizer defines a penalty if the source classifier and the target classifier make different predictions on an unlabeled target instance. However, with this regularizer, EA++ does not strictly restrict either the source classifier or the target classifier to lie in the target subspace $\mathcal{X}_{\mathrm{t}}$. In contrast, as we have pointed out above, when only the induced features are used, our method leverages the unlabeled target instances to force the learned classifier to lie in $\mathcal{X}_{\mathrm{t}}$.

3 Experiments

3.1 Tasks and Data Sets

We consider the following NLP tasks.

Personalized Spam Filtering (Spam): The data set comes from ECML/PKDD 2006 discovery challenge. The goal is to adapt a spam filter trained on a common pool of 4000 labeled emails to three individual users' personal inboxes, each containing 2500 emails. We use bag-of-word features for this task, and we report classification accuracy.

Gene Name Recognition (NER): The data set comes from BioCreAtIvE Task 1B (Hirschman et al., 2005). It contains three sets of Medline abstracts with labeled gene names. Each set corresponds to a single species (fly, mouse or yeast). We consider domain adaptation from one species to another. We use standard NER features including words, POS tags, prefixes/suffixes and contextual features. We report F1 scores for this task.

Relation Extraction (Relation): We use the ACE2005 data where the annotated documents are from several different sources such as broadcast news and conversational telephone speech. We report the F1 scores of identifying the 7 major relation types. We use standard features including entity types, entity head words, contextual words and other syntactic features derived from parse trees.

3.2 Methods for Comparison

Naive uses the original features.

Common uses only features commonly seen in both domains.

SCL is our implementation of Structural Correspondence Learning (Blitzer et al., 2006). We set the number of induced features to 50 based on preliminary experiments. For pivot features, we follow the setting used by Blitzer et al. (2006) and select the features with a term frequency more than 50 in both domains.

PCA uses principal component analysis on D^{t} to obtain K-dimensional induced feature vectors and then appends them to the original feature vectors.

ISF is our method using instance similarity features. We first transform each training instance to a K-dimensional vector according to Equation 1 and then append the vector to the original vector.

For all the three NLP tasks and the methods above that we compare, we employ the logistic regression (a.k.a. maximum entropy) classification algorithm with L_2 regularization to train a classifier, which means the loss function is the cross entropy error. We use the L-BFGS optimization algorithm to optimize our objective function.

3.3 Results

In Table 2, we show the comparison between our method and Naive, Common and SCL. For ISF, the parameter K is set to 100 for Spam, 50 for NER and 500 for Relation after tuning. As we can see from the table, Common, which removes source domain specific features during training, can sometimes improve the classification performance, but this is not consistent and the improvement is small. SCL can improve the performance in most settings for all three tasks, which confirms the general effectiveness of this method. For our method ISF, we can see that on average it outperforms both Naive and SCL significantly. When we zoom into the different source-target domain pairs of the three tasks, we can see that ISF outperforms SCL in most of the cases. This shows that our method is competitive despite its simplicity. It is also worth pointing out that SCL incurs much more computational cost than ISF.

We next compare ISF with PCA. Because PCA is also expensive, we only managed to run it on the Spam task. Table 3 shows that ISF also outperforms PCA significantly.

Method	**Spam**				**NER**							**Relation**
	u00	u01	u02	**average**	f→y	f→m	m→y	m→f	y→f	y→m	**average**	**average**
Naive	0.678	0.710	0.816	0.735	0.396	0.379	0.526	0.222	0.050	0.339	0.319	0.398
Common	0.697	0.732	0.781	0.737	0.409	0.388	0.559	0.208	0.059	0.344	0.328	0.401
SCL	0.699	0.717	0.824	0.747	0.405	0.380	0.525	**0.239**	0.063	0.35	0.327	0.403
ISF	**0.720**	**0.769**	**0.884**	**0.791****	**0.415**	**0.395**	**0.566**	0.212	**0.079**	**0.360**	**0.338****	**0.416****

Method	**Relation**									
	bc→bn	bc→cts	bc→nw	bc→un	bc→wl	bn→bc	bn→cts	bn→nw	bn→un	bn→wl
Naive	0.455	0.400	0.445	0.376	0.397	0.528	0.430	0.482	0.469	0.454
Common	**0.484**	0.408	0.446	0.373	0.400	0.536	0.452	0.478	0.465	0.444
SCL	0.467	0.395	0.453	0.391	**0.415**	0.531	0.434	**0.484**	0.461	**0.461**
ISF	0.474	**0.434**	**0.455**	**0.446**	0.405	**0.537**	**0.454**	**0.484**	**0.504**	0.460

	cts→bc	cts→bn	cts→nw	cts→un	cts→wl	nw2bc	nw→bn	nw→cts	nw→un	nw→wl
Naive	0.358	0.355	0.307	0.446	0.358	0.476	0.433	0.360	0.394	0.420
Common	0.345	0.336	0.292	0.432	0.339	0.475	**0.441**	**0.363**	0.399	0.429
SCL	0.361	0.359	0.314	0.448	0.357	0.480	0.439	0.354	**0.405**	0.426
ISF	**0.387**	**0.377**	**0.333**	**0.449**	**0.361**	**0.488**	0.439	0.342	0.401	**0.431**

	un→bc	un→bn	un→cts	un→nw	un→wl	wl→bc	wl→bn	wl→cts	wl→nw	wl→un
Naive	0.373	0.394	0.423	0.357	0.375	0.355	0.338	0.282	0.373	0.316
Common	0.399	**0.409**	0.416	0.370	0.370	0.351	0.364	**0.298**	0.379	**0.335**
SCL	0.379	0.399	0.423	0.356	0.377	0.361	0.355	0.288	0.381	0.330
ISF	**0.442**	0.404	**0.436**	**0.381**	**0.380**	**0.389**	**0.368**	**0.298**	**0.395**	0.329

Table 2: Comparison of performance on three NLP tasks. For each source-target pair of each task, the performance shown is the average of 5-fold cross validation. We also report the overall average performance for each task. We tested statistical significance only for the overall average performance and found that ISF was significantly better than both Naive and SCL with $p < 0.05$ (indicated by **) based on the Wilcoxon signed-rank test.

Method	**Spam**			
	u00	u01	u02	**average**
Naive	0.678	0.710	0.816	0.735
PCA	0.700	0.718	0.818	0.745
ISF	**0.720**	**0.769**	**0.884**	**0.791****

Table 3: Comparison between ISF and PCA.

4 Conclusions

We presented a hassle-free unsupervised domain adaptation method. The method is simple to implement, fast to run and yet effective for a few NLP tasks, outperforming SCL, a widely-used unsupervised domain adaptation method. We believe the proposed method can benefit a large number of practitioners who prefer simple methods than sophisticated domain adaptation methods.

Acknowledgment

We would like to thank the reviewers for their valuable comments.

References

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics.

John Blitzer, Sham Kakade, and Dean P. Foster. 2011. Domain adaptation with coupled subspaces. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 173–181.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 49–56.

Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*.

Hal Daume III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 407–412.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of* *the Association of Computational Linguistics*, pages 256–263.

Mark Dredze and Koby Crammer. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 689–697.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In Proceedings of the Twenty-eight International Conference on Machine Learning*.

Lynette Hirschman, Marc Colosimo, Alexander Morgan, and Alexander Yeh. 2005. Overview of BioCreAtIvE task 1B: normailized gene lists. *BMC Bioinformatics*, 6(Suppl 1):S11.

Yuening Hu, Ke Zhai, Vladimir Eidelman, and Jordan Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1166–1176.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271.

Abhishek Kumar, Avishek Saha, and Hal Daume. 2010. Co-regularization based semi-supervised domain adaptation. In *Advances in neural information processing systems*, pages 478–486.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 68–74.

Minh Luan Nguyen, Ivor W. Tsang, Kian Ming A. Chai, and Hai Leong Chieu. 2014. Robust domain adaptation for relation extraction via clustering consistency. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 807–817.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Chensheng Sun and Kin-Man Lam. 2013. Multiple-kernel, multiple-instance similarity features for efficient visual object detection. *IEEE Transactions on Image Processing*, 22(8):3050–3061.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 62–71.

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 538–544.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 672–682.

Pei Yang, Wei Gao, Qi Tan, and Kam-Fai Wong. 2012. Information-theoretic multi-view domain adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 270–274.