Transfer Joint Embedding for Cross-Domain Named Entity Recognition

SINNO JIALIN PAN, ZHIQIANG TOH, and JIAN SU, Institute for Infocomm Research, Singapore

Named Entity Recognition (NER) is a fundamental task in information extraction from unstructured text. Most previous machine-learning-based NER systems are domain-specific, which implies that they may only perform well on some specific domains (e.g., *Newswire*) but tend to adapt poorly to other related but different domains (e.g., *Weblog*). Recently, transfer learning techniques have been proposed to NER. However, most transfer learning approaches to NER are developed for binary classification, while NER is a multiclass classification problem in nature. Therefore, one has to first reduce the NER task to multiple binary classification tasks and solve them independently. In this article, we propose a new transfer learning method, named *Transfer Joint Embedding* (TJE), for cross-domain multiclass classification, which can fully exploit the relationships between classes (labels), and reduce domain difference in data distributions for transfer learning. More specifically, we aim to embed both labels (outputs) and high-dimensional features (inputs) from different domains (e.g., a source domain and a target domain) into a unified low-dimensional latent space, where 1) each label is represented by a prototype and the intrinsic relationships between labels can be measured by Euclidean distance; 2) the *distance* in data distributions between the source and target domains can be reduced; 3) the source domain labeled data are closer to their corresponding label-prototypes than others. After the latent space is learned, classification on the target domain data can be done with the simple nearest neighbor rule in the latent space. Furthermore, in order to scale up TJE, we propose an efficient algorithm based on stochastic gradient descent (SGD). Finally, we apply the proposed TJE method for NER across different domains on the ACE 2005 dataset, which is a benchmark in Natural Language Processing (NLP). Experimental results demonstrate the effectiveness of TJE and show that TJE can outperform state-of-the-art transfer learning approaches to NER.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Data Mining

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Named entity recognition, transfer learning, multiclass classification

ACM Reference Format:
Pan, S. J., Toh, Z., and Su, J. 2013. Transfer joint embedding for cross-domain named entity recognition. ACM Trans. Inf. Syst. 31, 2, Article 7 (May 2013), 27 pages.
DOI: http://dx.doi.org/10.1145/2457465.2457467

1. INTRODUCTION

Information extraction, which aims to automatically extract structured information from unstructured or semi-structured text or Web pages, is an important technology for many applications, such as information retrieval [Manning et al. 2008], question answering [Kwok et al. 2001], financial analysis [Schumaker and Chen 2009], etc. Named Entity Recognition (NER) is one of the fundamental tasks in information extraction. The objective of NER is to identify and classify every word/term in text into predefined

Author's address: Data Analytics Department, Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632; email: {jspan, ztoh, sujian}@i2r.a-star.edu.sg.

DOI: http://dx.doi.org/10.1145/2457465.2457467

types of entities, such as persons, organizations, locations, etc., or "none-of-the-above". Most current NER systems are based on machine learning techniques, which reply on a lot of labeled data for training predictive models. Usually, the labeled training data contain documents and the corresponding tags (e.g., persons, organizations, or "none-of-the-above") to each term/word within the documents. Buidling models is an expensive process requiring tedious effort on annotating training data. Furthermore, due to different writing styles and lexicons used in different domains, a NER system trained on one domain, for instance, *Newswire*, may not perform well on another domain, for instance, *Weblog*. As a result, adaptation techniques are highly desirable for NER such that a NER system trained on one domain can adapt well to other domains with little extra supervision. Transfer learning, which aims to enable systems to extract and apply knowledge learned from previous tasks to a novel task, is such a potential technology for cross-domain NER.

In the past few years, transfer learning has been studied in NER to reduce labeling effort across different domains. Daumé III [2007] proposed a simple feature augmentation method for NER. However, this method requires some labeled data available in the target domain. In many real-world scenarios, it is more desirable if no additional labeling effort is required to adapt NER systems across domains. Therefore, in this article, we focus on the setting where there are no labeled data but a lot of unlabeled data (documents without any tags to words/items) in the target domain for adaptation. In this setting, several transfer learning methods have been proposed to NER [Jiang and Zhai 2007; Wu et al. 2009]. These methods are based on bootstrapping, which can be referred to as instance-based transfer learning methods [Pan and Yang 2010]. However, for cross-domain NER, the percentage of overlapping features across domains can be very small. In this case, instance-based transfer learning methods may not work well. Furthermore, these methods are developed for binary classification, requiring multiple binary classifiers for each type of entities independently. Therefore, they may suffer from 1) the relationships between different types of entities cannot be fully exploited, and 2) when the number of entity-types is huge, the computational cost is expensive.

In this article, we propose a new transfer learning method for cross-domain multiclass classification with application to NER.[1] Specifically, we first embed labels (or classes) into a low-dimensional latent space $\mathcal{F}$. In this case, the relationships between labels, for instance, similar or dissimilar, can be measured by the distance between their corresponding label-prototypes in the latent space, for instance, close to or far away from each other. Meanwhile, we embed high-dimensional inputs (features) from the source and target domains into another low-dimensional latent space $\mathcal{F}'$, projected onto which the distance between the source and target domain data can be reduced and the original data structure can be preserved. Finally, a mapping is learned from the new feature space $\mathcal{F}'$ to the label latent space $\mathcal{F}$, such that the projected instances are closer to their corresponding label-prototypes than others. In this way, classification on unseen target domain data can be done with the simple nearest neighbor rule in the label latent space. Furthermore, we propose two solutions based on different optimization strategies. Note that, though in this article we use cross-domain NER

[1]Note that some researchers have shown that context information within sentences is useful for NER and proposed to use sequential labeling methods for NER [Zhou and Su 2002; Finkel et al. 2005]. However, more recently, some other researchers pointed out that the sequential labeling models may suffer from inability on modeling *nested* named entities [Finkel and Manning 2009]. In this article, we focus on the cross-domain problem and consider NER as a multiclass classification task instead of sequential labeling. In deed, based on the outputs of a multiclass classifier, one can postuse a Vierbi parse [Rabiner and Juang 1986] to find the valid sequence of entity types with the highest probability to achieve better performance.

as an evaluation application, the proposed method is general for other cross-domain multiclass classification applications.[2]

The rest of the article is organized as follows. In the following section, we first review some related work and introduce preliminaries that are used in our proposed method. In Section 3, we describe the problem statement and give an overview of the proposed method. In Section 4, we present the proposed method, named *Transfer Joint Embedding* (TJE), in detail. In Section 5, we demonstrate the effectiveness of TJE through expensive experiments on the ACE 2005 dataset. Finally, in Section 6, we conclude this work and discuss some future directions.

2. RELATED WORK AND PRELIMINARIES

2.1. Named Entity Recognition

Generally speaking, Named Entity Recognition (NER) consists of two subtasks [Nadeau and Sekine 2007]: 1) Named Entity identification, and 2) Named Entity classification. Named Entity identification is to identify whether a word/term is a named entity or not, which can be referred to as a binary classification task. For the terms/words that are already identified as entities, the goal of Named Entity Classification is to classify them into predefined categories (different types of entities, e.g., persons, organizations or locations, etc.). If we consider the tag "none-of-the-above" as an additional class, then NER can be referred to as a unified multiclass classification problem, whose goal is to classify each word/term into predefined types of entities or "none-of-the-above".

Earlier NER systems mainly employed rule-based approaches such as handcrafted rules and finite state patterns, such as the TLG system [Mikheev et al. 1998, 1999], the SRA system [Aone et al. 1998], the LaSIE-II system [Humphreys et al. 1998] and the NetOwl system [Krupka and Hausman 1998]. Rule-based NER systems can perform well when the human-decided rules match the scenarios perfectly, but they are hard to handle uncertainty. As annotated corpora become available, machine-learning approaches, such as Hidden Markov Models [Zhou and Su 2002], Support Vector Machines (SVMs) [Isozaki and Kazawa 2002] and Conditional Random Fields [Finkel et al. 2005], have been proposed to train classifiers or models from the annotated corpora for NER automatically. Recently, machine-learning-based NER systems have shown their success in NER, but most of them suffer from two main limitations: 1) the performance of NER systems relies on a large number of labeled data for training, and 2) most systems are domain-specific, which implies that a NER trained on one domain tends to perform badly on other domains [Ciaramita and Altun 2005; Jiang and Zhai 2006]. In order to reduce the effect for building NER systems on new domains, some researchers have proposed to use transfer learning techniques to utilize labeled data or extract knowledge from existing NER systems.

2.2. Transfer Learning

In Natural Language Processing (NLP), transfer learning can be referred to as domain adaptation [Pan and Yang 2010]. Previous work in domain adaptation can be classified into two settings: 1) a small amount of labeled data are available in the target domain [Daumé III 2007; Blitzer et al. 2007; Jiang and Zhai 2007], and 2) no labeled data but some unlabeled data are available in the target domain [Blitzer et al. 2006; Jiang and Zhai 2007; Wu et al. 2009; Pan et al. 2008; Pan et al. 2011; Pan et al. 2010; Glorot et al. 2011]. In this article, we focus on the latter setting, which is

[2]Applying the proposed method to other cross-domain multiclass classification applications is one of our future directions.

more practical in real-world applications. In this setting, existing domain adaptation approaches can be further classified into two categories: instance-based and feature-based approaches [Pan and Yang 2010]. Instance-based approaches assume that part of labeled data from the source domain are useful to build models for the target domain after reweighting. So far, many instance reweighting techniques have been proposed to domain adaptation, such as bootstrapping [Jiang and Zhai 2007] and correcting sample selection bias or covariate shift techniques [Quionero-Candela et al. 2009]. Different from instance-based approaches, feature-based approaches assume that there exists a *good* feature representation across the source and target domains [Ben-David et al. 2007]. Based on the feature representation, domain difference can be reduced, and many existing models can be reused for cross-domain classification. Some methods have been proposed for uncovering such good feature representation, such as structural correspondence learning (SCL) [Blitzer et al. 2006], maximum mean discrepancy embedding (MMDE) [Pan et al. 2008], transfer component analysis (TCA) [Pan et al. 2011], spectral feature alignment (SFA) [Pan et al. 2010], and deep-learning-based feature learning [Glorot et al. 2011].

For domain adaptation in NER, Jiang and Zhai [2007] proposed the Balanced Bootstrapping (BB) algorithm. The goal of BB is to iteratively select most confidently labeled data from the target domain and add them to the source domain training pool to train a more precise model for the target domain. Wu et al. [2009] proposed another bootstrapping algorithm named Domain Adaptive Bootstrapping (DAB) for NER. In DAB, new instance selection strategy and termination criterion are proposed. Experimental results showed that DAB can outperform the standard bootstrapping algorithm and BB in cross-domain NER. However, neither BB nor DAB can work for multiclass classification problems naturally. By using BB or DAB, one has to first reduce a NER task into multiple binary classification tasks using the *one-vs-rest* strategy, and then train multiple binary classifiers independently. Finally, predictions are made according to the outputs of all binary classifiers. As a result, either BB or DAB fails to fully exploit the relationships between different types of entities, which indeed can be used to further boost the performance of NER. Daumé III [2007] proposed a simple feature augmentation method for NER. This method can train multiclass models on the augmented features naturally. However, it requires some labeled data available in the target domain, which is beyond the setting discussed in this work. More recently, Chen et al. [2009] proposed a feature-based domain adaptation method for multiclass text mining, which aims to learn a subspace to reduce domain difference and capture label dependency. However, this method still needs to train multiple binary classifiers using the *one-vs-rest* strategy. Furthermore, in their proposed method, for each classifier, an iterative domain adaptation approach is performed on the source and target domain data, which requires eigendecomposition on a $m \times m$ matrix in each iteration, where m is the number of original features. This may become computational expensive in both training and testing stages when both the numbers of classes and features are large [Whitelaw et al. 2008].

Different from previous methods, in this article, we propose a new transfer learning method for NER, where we employ the label embedding techniques [Weinberger and Chapelle 2009; Bengio et al. 2010] to transform multiclass classification to regression in a low-dimensional latent space. Meanwhile, we employ the feature-based domain adaptation techniques [Pan et al. 2008, 2011] to ensure that the distance in data distributions between the source and target domains is minimized in the latent space. In this way, the relationships between classes (types of entities) can be fully exploited and difference between domains can be reduced in the latent space. Furthermore, we propose two solutions to solve the objective efficiently even when the number of classes is large.

2.3. Notation

In the sequel, matrices and vectors are both written in bold (e.g., $\mathbf{X}$, $\mathbf{x}$). $\mathbf{1}$ is the column vector with all ones, and $\mathbf{I}$ is the identity matrix. The transpose of vector or matrix is denoted by the superscript $^\top$, $\mathrm{tr}(\mathbf{X})$ and $\mathbf{X}^{-1}$ denote the trace and inverse of $\mathbf{X}$ respectively, and $\|\mathbf{X}\|_F$ denotes the Frobenius norm of $\mathbf{X}$. As mentioned in the previous section, in this article, we focus on the transfer learning setting where sufficient labeled data $\mathcal{D}_S$ are available in the source domain and only unlabeled data D_T are available in the target domain. Let the source domain data be $\mathcal{D}_S = \{(\mathbf{x}_{S_i}, y_{S_i})\}_{i=1}^{n_1}$, where $\mathbf{x}_{S_i} \in \mathcal{X} \subseteq \mathbb{R}^{1\times m}$ is the input and $y_{S_i} \in \mathcal{Y}$ is the corresponding output. Similarly, let the target domain data be $\mathcal{D}_T = \{\mathbf{x}_{T_i}\}_{j=1}^{n_2}$, where the input $\mathbf{x}_{T_i}$ is also in $\mathcal{X}$. We assume the source and target domain share the same label space $\mathcal{Y}$. Let $\mathcal{P}(\mathbf{X}_S)$ and $\mathcal{P}(\mathbf{X}_T)$ be the marginal distributions of $\mathbf{X}_S = \{\mathbf{x}_{S_i}\}$ and $\mathbf{X}_T = \{\mathbf{x}_{T_i}\}$ in the source and target domains respectively. In general, $\mathcal{P}(\mathbf{X}_S)$ and $\mathcal{P}(\mathbf{X}_T)$ can be different. Our task is to make predictions on unseen test data $\{\mathbf{x}^*_{T_i}\}$'s in the target domain. The key assumption in most domain adaptation methods is that $\mathcal{P}(\mathbf{X}_S) \neq \mathcal{P}(\mathbf{X}_T)$, but $\mathcal{P}(\mathbf{Y}_S|\mathbf{X}_S) = \mathcal{P}(\mathbf{Y}_T|\mathbf{X}_T)$. Through this article, we denote $\alpha, \beta \in \{1, \dots, c\}$ class or label indices, and $\phi(\alpha) = [0, \dots, 1, \dots, 0]$ a row vector of dimensionality c with all zeros but a single 1 in the α^{th} position, and use the words *label* and *class* interchangeably.

2.4. Hilbert Space Embedding of Distributions

2.4.1. Maximum Mean Discrepancy. Given samples $\mathbf{X} = \{\mathbf{x}_i\}$ and $\mathbf{Z} = \{\mathbf{z}_i\}$ drawn from two distributions, Maximum Mean Discrepancy (MMD) [Smola et al. 2007] is a nonparametric distance estimate for measuring their distance. Different from other criteria, such as the Kullback-Leibler (KL) divergence, MMD does not require an intermediate density estimate, and can estimate the distance between distributions in a Reproducing Kernel Hilbert Space (RKHS) directly [Gretton et al. 2007]. Let the kernel-induced feature map be φ. The empirical estimate of MMD between $\{\mathbf{x}_1, \dots, \mathbf{x}_{n_1}\}$ and $\{\mathbf{z}_1, \dots, \mathbf{z}_{n_2}\}$ is

$$\mathrm{MMD}(\mathbf{X}, \mathbf{Z}) = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \varphi(\mathbf{x}_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \varphi(\mathbf{z}_i) \right\|_{\mathcal{H}}^2,$$

where $\|\cdot\|_{\mathcal{H}}$ is the RKHS norm. Therefore, the distance between two distributions is simply the distance between the two mean elements in a RKHS. It can be shown that when the RKHS is universal, MMD asymptotically approaches zero if and only if the two distributions are the same [Smola et al. 2007].

2.4.2. Hilbert-Schmidt Independence Criterion. Related to the MMD, the Hilbert-Schmidt Independence Criterion (HSIC) [Gretton et al. 2005] is a simple yet powerful nonparametric criterion for measuring the dependence between the sets $\mathbf{X}$ and $\mathbf{Y}$. As its name implies, it computes the Hilbert-Schmidt norm of a cross-covariance operator in the RKHS. An (biased) empirical estimate can be easily obtained from the corresponding kernel matrices, as

$$\mathrm{HSIC}(\mathbf{X}, \mathbf{Y}) = \frac{1}{(n-1)^2} \mathrm{tr}(\mathbf{HKHK}_y),$$

where $\mathbf{K}, \mathbf{K}_y$ are kernel matrices defined on $\mathbf{X}$ and $\mathbf{Y}$ respectively, $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix and n is the number of samples in $\mathbf{X}$ and $\mathbf{Y}$. Similar to MMD, it can also be shown that if the RKHS is universal, HSIC asymptotically approaches zero if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent. Conversely, a large HSIC value suggests strong dependence.

2.5. Domain Adaptation via MMD and HSIC

Recently, Pan et al. [2011] proposed a semi-supervised feature extraction method named Semi-Supervised Transfer Component Analysis (SSTCA) based on the MMD and HSIC criteria for domain adaptation. The motivation behind SSTCA is that there may exist some common latent factors between the source and target domains, projected onto which the domain difference can be reduced and original data structure (e.g., data variance, local geometric structure, dependency to side information, etc.) can be preserved. As a result, the latent space spanned by the latent factors can be used as a bridge for cross-domain classification or regression tasks. In summary, SSTCA tries to learn a mapping φ to map the source and target domain data to a latent space, where the distance in data distributions between the source and target domains measured by $\mathrm{MMD}(\mathcal{P}(\varphi(\mathbf{X}_S)), \mathcal{P}(\varphi(\mathbf{X}_T)))$ is small, and the dependence between the embeddings and labels of the source domain data measured by $\mathrm{HSIC}(\varphi(\mathbf{X}_S), \mathbf{Y}_S)$ is large. By assuming that the kernel defined on inputs used in MMD and HSIC is linear, that is, $\varphi(\mathbf{x}) = \mathbf{x}\boldsymbol{\Theta}$, where $\boldsymbol{\Theta} \in \mathbb{R}^{m\times p}$, the goal of SSTCA is to learn $\boldsymbol{\Theta}$ by optimizing the following objectives simultaneously,

$$\min_{\boldsymbol{\Theta}} \ \mathrm{tr}(\mathbf{X}\boldsymbol{\Theta}\boldsymbol{\Theta}^\top\mathbf{X}^\top\mathbf{L}) + \mu\ \mathrm{tr}(\boldsymbol{\Theta}^\top\boldsymbol{\Theta}), \tag{1}$$

$$\max_{\boldsymbol{\Theta}} \ \frac{1}{n_1^2}\mathrm{tr}\big(\mathbf{H}\mathbf{X}_S\boldsymbol{\Theta}\boldsymbol{\Theta}^\top\mathbf{X}_S^\top\mathbf{H}\mathbf{K}_y\big), \tag{2}$$

where $\mathbf{X} = [\mathbf{X}_S^\top\ \mathbf{X}_T^\top]^\top$, $\mathbf{L}_{ij} = 1/n_1^2$ if $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_S$; $\mathbf{L}_{ij} = 1/n_2^2$ if $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_T$; otherwise, $\mathbf{L}_{ij} = -1/(n_1n_2)$, $\mathbf{K}_y$ is a kernel matrix defined on labels $\mathbf{Y}_S$, and $\mu \geq 0$ is a tradeoff parameter. The first term in the minimization objective is to reduce the distance in data distributions between the source and target domains, the second term is a regularization term, while the maximization objective is to maximize the dependence between the embedding and labels. To solve the two objectives simultaneously, Pan et al. [2011] proposed the following constrained minimization problem,

$$\begin{aligned}&\min_{\boldsymbol{\Theta}} \ \mathrm{tr}(\mathbf{X}\boldsymbol{\Theta}\boldsymbol{\Theta}^\top\mathbf{X}^\top\mathbf{L}) + \mu\ \mathrm{tr}(\boldsymbol{\Theta}^\top\boldsymbol{\Theta})\\ &s.t.\ \ \mathrm{tr}\big(\mathbf{H}\mathbf{X}_S\boldsymbol{\Theta}\boldsymbol{\Theta}^\top\mathbf{X}_S^\top\mathbf{H}\mathbf{K}_y\big) = \mathbf{I}.\end{aligned} \tag{3}$$

By using the method of Lagrange multipliers, it can be shown that the above optimization problem can be solved by eigen-decomposing on the following $m \times m$ matrix [Pan et al. 2011],

$$(\mathbf{X}^\top\mathbf{L}\mathbf{X} + \mu\ \mathbf{I})^{-1}\mathbf{X}_S^\top\mathbf{H}\mathbf{K}_y\mathbf{H}\mathbf{X}_S. \tag{4}$$

The computational time is $O(p \times (n_1 + n_2)^2)$ when p nonzero eigenvectors are extracted.

3. PROBLEM STATEMENT AND OVERALL APPROACH

Recall that we have no labeled data in the target domain, while we have plenty of labeled data $\mathcal{D}_S = \{(\mathbf{x}_{S_i}, y_{S_i})\}_{i=1}^{n_1}$ in the source domain and some unlabeled data $\mathcal{D}_T = \{\mathbf{x}_{T_j}\}_{j=1}^{n_2}$ in the target domain. In domain adaptation, $\mathcal{P}(\mathbf{X}_S)$ and $\mathcal{P}(\mathbf{X}_T)$ may not be the same. Our task is to make predictions on unseen test data $\mathcal{D}_T^* = \{\mathbf{x}_{T_i}^*\}_{i=1}^{n}$ in the target domain.

Our proposed Transfer Joint Embedding (TJE) consists of the following three components.

(1) For labels, we aim to find a mapping $\boldsymbol{\Phi}_y$ to embed each label $\alpha \in \mathcal{Y}$ into a label latent space $\mathcal{F}$ of dimensionality q. The new representation $\boldsymbol{\Phi}_y(\alpha)$ can be referred to as the prototype of the label α in the latent space. Intuitively, if two labels α and

β are related, then the distance between their corresponding prototypes $\mathbf{\Phi}_y(\alpha)$ and $\mathbf{\Phi}_y(\beta)$ is small, otherwise, far away.

(2) For reducing the difference between domains, we aim to discover an adaptive mapping $\mathbf{\Phi}_a$ to project the source and target domain data onto an intermediate feature latent space $\mathcal{F}'$ of dimensionality p, where the distance between data distributions of the source and target domains $\mathcal{P}(\mathbf{\Phi}_a(\mathbf{X}_S))$ and $\mathcal{P}(\mathbf{\Phi}_a(\mathbf{X}_T))$ is small.

(3) Finally, we need to learn a mapping $\mathbf{\Phi}_x$ to map the adapted representations $\mathbf{\Phi}_a(\mathbf{X}_S)$ and $\mathbf{\Phi}_a(\mathbf{X}_T)$ from $\mathcal{F}'$ to $\mathcal{F}$, such that in the label latent space $\mathcal{F}$, the adapted source domain instances $\mathbf{\Phi}_x(\mathbf{\Phi}_a(\mathbf{X}_S))$ are closer to their corresponding label-prototypes $\mathbf{\Phi}_y(\mathbf{Y}_S)$ than others.

After we obtain $\mathbf{\Phi}_y$, $\mathbf{\Phi}_a$ and $\mathbf{\Phi}_x$, for any unseen data $\mathbf{x}_T^*$ from the target domain, we can employ the following nearest neighbor classification rule to predict its label.

$$y_T^* = \arg\min_{\alpha} \left\| \mathbf{\Phi}_x\left(\mathbf{\Phi}_a\left(\mathbf{x}_T^*\right)\right) - \mathbf{\Phi}_y(\alpha) \right\|_2 .$$

4. TRANSFER JOINT EMBEDDING

In this work, we assume that $\mathbf{\Phi}_y$, $\mathbf{\Phi}_a$ and $\mathbf{\Phi}_x$ are linear mappings and can be written as $\mathbf{\Phi}_y(\alpha) = \phi(\alpha)\mathbf{V}$, $\mathbf{\Phi}_a(x) = x\mathbf{\Theta}$ and $\mathbf{\Phi}_x(\widetilde{x}) = \widetilde{x}\mathbf{W}$, respectively, where $\mathbf{V} \in \mathbb{R}^{c\times q}$, $\mathbf{\Theta} \in \mathbb{R}^{m\times p}$ and $\mathbf{W} \in \mathbb{R}^{p\times q}$. We present two algorithms to learn the mappings $\mathbf{\Phi}_y$, $\mathbf{\Phi}_a$ and $\mathbf{\Phi}_x$ ($\mathbf{V}$, $\mathbf{\Theta}$ and $\mathbf{W}$ in particular). One is based on sequential-optimization and the other is based on joint-optimization.

4.1. Label Embedding

The first step is to embed the labels into a low-dimensional latent space. By assuming $\Phi_y(\alpha) = \phi(\alpha)\mathbf{V}$, our goal is to learn the mapping $\mathbf{V} \in \mathbb{R}^{c\times q}$ for labels. There are various ways to learn $\mathbf{V}$. The simplest method is to set $\mathbf{V} = \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{c\times c}$ denotes the identity matrix. In this case, $q = c$ and the label-prototypes are all in distance $\sqrt{2}$ from each other at the corner of a $(c-1)$-dimensional simplex. However, in many real-world applications, some labels may be more related to each other. As a result, the distance between their corresponding prototypes tend to be smaller. Furthermore, when c is huge, it would be more desirable to pick $q \ll c$.

An alterative approach is to derive the mapping $\mathbf{V}$ for the label-prototypes from a cost matrix $\mathbf{C} \in \mathbb{R}^{c\times c}$, which is a dissimilarity matrix defined on the labels, by using metric multidimensional scaling (MDS) [Cox and Cox 1994; Weinberger and Chapelle 2009]. However, in general, a cost matrix $\mathbf{C}$ is not always available or cannot be properly defined. Therefore, in this article, we borrow the idea from Bengio et al. [2010] to construct the mapping $\mathbf{V}$ using the following method. First, we train independent classifiers f_i, for instance, Support Vector Machines (SVMs), for each label $1, \ldots, c$, and compute the $c \times c$ confusion matrix $\mathbf{M}$ over the source domain labeled data $\mathcal{D}_S$. By considering the confusion matrix $\mathbf{M}$ as a similarity matrix, we then compute its corresponding Laplacian matrix $\mathcal{L} = \mathbf{D} - \mathbf{M}$, where $\mathbf{D}_{ii} = \sum_j \mathbf{M}_{ij}$. Finally, we recover the label-embedding mapping $\mathbf{V}$ by using Laplacian Eigenmaps [Belkin and Niyogi 2003] as follows,

$$\begin{aligned} \min_{\mathbf{V}} \;& \operatorname{tr}(\mathbf{V}^\top \mathcal{L}\mathbf{V}) \\ \text{s.t.} \;& \mathbf{V}^\top\mathbf{D}\mathbf{V} = \mathbf{I}. \end{aligned} \qquad (5)$$

Note that the optimal solution of the above optimization problem can be obtained by solving a generalized eigen-decomposition problem. Here, we assume that though the data distributions may differ across domains, the confusion matrices over labels in different domains are similar.

4.2. Transfer Feature Embedding

After we find a suitable mapping $\mathbf{V}$ to embed the labels into a latent space of dimensionality q, the problem of multiclass classification has been transformed to a q-variate regression problem. However, so far we have not reduced the difference between the source and target domains, resulting in that a regression model trained on the source domain may still not be able to make precise predictions on the target domain. Therefore, we need to look for a new data representation to reduce the domain difference. We can apply SSTCA as introduced in (3) to learn a feature mapping $\mathbf{\Theta}$, such that the distance in distributions between the mapped source and target domain data, $\text{Dist}(\mathcal{P}(\mathbf{X}_S\mathbf{\Theta}), \mathcal{P}(\mathbf{X}_T\mathbf{\Theta}))$, is minimized and $\mathbf{X}_S\mathbf{\Theta}$ is relevant to $\phi(\mathbf{Y}_S)\mathbf{V}$. As will be introduced in the joint-optimization solution, we aims to use an iterative algorithm to solve $\mathbf{\Theta}$ for large-scale data. Therefore, to avoid the expensive computation on performing inverse and eigen-decomposition on the $m \times m$ matrix (4) in each iteration, we modify the original objective of SSTCA to the following unconstrained optimization problem, which can be solved by either eigen-decomposition or gradient-descent methods,

$$\min_{\mathbf{\Theta}} \text{tr}\left(\mathbf{X}\mathbf{\Theta}\mathbf{\Theta}^\top\mathbf{X}^\top\mathbf{L} - \frac{1}{n_1^2}\mathbf{H}\mathbf{X}_S\mathbf{\Theta}\mathbf{\Theta}^\top\mathbf{X}_S^\top\mathbf{H}\mathbf{K}_y + \mu\ \mathbf{\Theta}^\top\mathbf{\Theta}\right), \tag{6}$$

where we use the linear kernel for *labels*, $\mathbf{K}_y = \mathbf{\Phi}_y(\mathbf{Y}_S)\mathbf{\Phi}_y(\mathbf{Y}_S)^\top = (\phi(\mathbf{Y}_S)\mathbf{V})(\phi(\mathbf{Y}_S)\mathbf{V})^\top$, and $\mathbf{X}$, $\mathbf{H}$ and $\mathbf{L}$ are the matrices as defined in (1). The objective (6) can be further rewritten as

$$\min_{\mathbf{\Theta}} \text{tr}\left(\mathbf{\Theta}^\top\left(\mathbf{\Upsilon}_1 + \mu\ \mathbf{I}\right)\mathbf{\Theta}\right), \tag{7}$$

where $\mathbf{\Upsilon}_1 = \mathbf{X}^\top\mathbf{L}\mathbf{X} - \frac{1}{n_1^2}\mathbf{X}_S^\top\mathbf{H}\phi(\mathbf{Y}_S)\mathbf{V}\mathbf{V}^\top\phi(\mathbf{Y}_S)^\top\mathbf{H}\mathbf{X}_S$. It can be proven that the optimal solution $\mathbf{\Theta}$ of (7) can be solved by eigen-decomposing on the matrix $\mathbf{\Upsilon}_1 + \mu\ \mathbf{I}$.

4.3. Classification through Regression in the Latent Space

So far, we have learned the embedding mapping $\mathbf{V} : \mathcal{Y} \to \mathcal{F}$ to map the labels to the latent space $\mathcal{F}$, and the domain adaptation mapping $\mathbf{\Theta} : \mathcal{X} \to \mathcal{F}'$ to project the source and target domain instances $\mathbf{X}_S$ and $\mathbf{X}_T$ onto the feature latent space $\mathcal{F}'$. All that remains is to learn the mapping $\mathbf{W} : \mathcal{F}' \to \mathcal{F}$ to map the adapted source domain feature vectors $\mathbf{X}_S\mathbf{\Theta}$ to their corresponding labels $\phi(\mathbf{Y}_S)\mathbf{V}$, such that source domain instances are close to the their corresponding label-prototypes, and source domain instances with different labels are well separated. Based on this objective, we formulate the following optimization problem to learn the mapping $\mathbf{W}$,

$$\min_{\mathbf{W}} h(\mathbf{W}) = \frac{1}{2n_1}\sum_{i=1}^{n_1}\left\|\phi(y_{S_i})\mathbf{V} - \mathbf{x}_{S_i}\mathbf{\Theta}\mathbf{W}\right\|_2^2 + \frac{\lambda_1}{2}\|\mathbf{W}\|_F^2 \tag{8}$$

where $\lambda_1 > 0$ is the parameter of the regularization term $\|\mathbf{W}\|_F$, which aims to avoid potential overfitting. It can be proven that the optimization (8) has the closed form solution,

$$\mathbf{W} = \frac{1}{n_1}\left(\frac{1}{n_1}\mathbf{\Theta}^\top\mathbf{X}_S^\top\mathbf{X}_S\mathbf{\Theta} + \lambda_1\mathbf{I}\right)^{-1}\mathbf{\Theta}^\top\mathbf{X}_S^\top\mathbf{J}_S\mathbf{V}, \tag{9}$$

where $\mathbf{J}_S \in \{0, 1\}^{n_1\times c}$ is an indexing matrix with $\mathbf{J}_{S_{i\alpha}} = 1$ if and only if $y_i = \alpha$. Note that instead of computing the matrix inverse $(\frac{1}{n_1}\mathbf{\Theta}^\top\mathbf{X}_S^\top\mathbf{X}_S\mathbf{\Theta} + \lambda_1\mathbf{I})^{-1}$, which may be expensive, (9) can be solved efficiently with linear conjugate gradient for each column of $\mathbf{W}$ independently.

ALGORITHM 1: Sequential optimization approach to Transfer Joint Embedding (TJE$_q$)

```
. Input: $\mathcal{D}_S$, $D_T$ and parameters: $\lambda_1$, $\mu$, $p$ and $q$.
. Output: $\mathbf{V}$, $\mathbf{\Theta}$ and $\mathbf{W}$.
1: Learn $\mathbf{V}$ by solving (5).
2: Learn $\mathbf{\Theta}$ by solving (7).
3: Learn $\mathbf{W}$ by using (9).
```

4.4. Prediction

The overall algorithm of the given sequential-optimization approach to learning $\mathbf{V}$, $\mathbf{\Theta}$ and $\mathbf{W}$ is described in Algorithm 1. After we obtain $\mathbf{V}$, $\mathbf{\Theta}$ and $\mathbf{W}$, for any unseen test data $\mathbf{x}_T^*$ from the target domain, we can employ the following rule to predict its labels.

$$y_T^* = \arg\min_{\alpha} \left\| \mathbf{x}_T^* \mathbf{\Theta} \mathbf{W} - \phi(\alpha)\mathbf{V} \right\|_2 . \tag{10}$$

Note that, using (10) for prediction requires to calculate the distances between the embedded test data $\mathbf{x}_T^*\mathbf{\Theta}\mathbf{W}$ and all label-prototypes $\phi(\alpha)\mathbf{V}$, $\alpha \in \{1, \ldots, c\}$. When c is huge, it would be computationally expensive. However, as proposed in Bengio et al. [2010], in the label embedding space $\mathcal{F}$, one can further learn a tree on labels to speed up prediction time.

4.5. A Unified Formulation

The sequential-optimization approach just described may not get optimal solutions for $\mathbf{V}$, $\mathbf{\Theta}$ and $\mathbf{W}$, since it tends to optimize the mappings independently. In this section, we present a unified optimization framework to learn the mappings jointly. Formally, we aim to solve the following objective,

$$\begin{aligned} \min_{\mathbf{V},\mathbf{\Theta},\mathbf{W}} \quad & \frac{1}{2n_1c} \sum_{i,\alpha} \xi_{i\alpha} + \frac{\lambda_1}{2}\|\mathbf{W}\|_F^2 + \frac{\lambda_2}{2}\operatorname{tr}\big(\mathbf{\Theta}^\top \mathbf{\Upsilon}_1 \mathbf{\Theta}\big) + \frac{\lambda_3}{2}\|\mathbf{\Theta}\|_F^2 \\ \text{s.t.} \quad & \left\|\phi(y_{S_i})\mathbf{V} - \mathbf{x}_{S_i}\mathbf{\Theta}\mathbf{W}\right\|_2^2 \le \left\|\phi(\alpha)\mathbf{V} - \mathbf{x}_{S_i}\mathbf{\Theta}\mathbf{W}\right\|_2^2 + \xi_{i\alpha}, \ \forall \alpha \in \mathcal{Y}\backslash y_{S_i} \\ & \xi_{i\alpha} \ge 0, \ i = 1, \ldots, n_1, \\ & \|\mathbf{V}_i\|_2 \le 1. \end{aligned} \tag{11}$$

where λ_1, λ_2 and λ_3 are nonnegative tradeoff parameters, $\mathbf{V}_i$ denotes the i^{th} row of $\mathbf{V}$, and $\mathbf{\Upsilon}_1$ is the same as defined in (7). The "soft" inequality constraints,

$$\left\|\phi(y_{S_i})\mathbf{V} - \mathbf{x}_{S_i}\mathbf{\Theta}\mathbf{W}\right\|_2^2 \le \left\|\phi(\alpha)\mathbf{V} - \mathbf{x}_{S_i}\mathbf{\Theta}\mathbf{W}\right\|_2^2 + \xi_{i\alpha},$$

are to enforce that for each source domain input $\mathbf{x}_{S_i}$, it is closer to its corresponding label-prototype y_{S_i} than other label-prototypes in the label latent space $\mathcal{F}$. The first term in the objective is to minimize the amount of violation of the "hard" inequality constraints absorbed by the slack-variables $\{\xi_{i\alpha} \ge 0\}$'s. The second and fourth terms in the objective are regularization terms on $\mathbf{W}$ and $\mathbf{\Theta}$ respectively. The third term aims to minimize the distance between the source and target domains after projection. The constraints $\|\mathbf{V}_i\|_2 \le 1$ are to control the scale of the label mapping $\mathbf{V}$. To solve the joint-optimization problem (11), we first reformulate the objective as

$$\begin{aligned} & \min \widetilde{h}(\mathbf{V}, \mathbf{\Theta}, \mathbf{W}) \\ & = \frac{1}{2n_1c}\sum_{i,\alpha} \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \mathbf{\Theta}, \mathbf{W}) + \frac{\lambda_1}{2}\|\mathbf{W}\|_F^2 + \frac{\lambda_2}{2}\operatorname{tr}\big(\mathbf{\Theta}^\top\mathbf{\Upsilon}_1\mathbf{\Theta}\big) + \frac{\lambda_3}{2}\|\mathbf{\Theta}\|_F^2, \\ & \text{s.t.} \ \|\mathbf{V}_i\|_2 \le 1, \end{aligned} \tag{12}$$

where the loss function $\ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \boldsymbol{\Theta}, \mathbf{W})$ is defined as

$$\ell(\cdot) = \max\left(0, \left\|\phi(y_{S_i})\mathbf{V} - \mathbf{x}_{S_i}\boldsymbol{\Theta}\mathbf{W}\right\|_2^2 - \left\|\phi(\alpha)\mathbf{V} - \mathbf{x}_{S_i}\boldsymbol{\Theta}\mathbf{W}\right\|_2^2\right), \tag{13}$$

which is similar to the *hinge-loss* in SVMs. The objective (12) is nonconvex with respect to $\mathbf{V}$, $\boldsymbol{\Theta}$ and $\mathbf{W}$, but it is convex with respect to any one of them when fixing the other two. As a result, we can fix any two matrices and apply subgradient descent methods to optimize the left one iteratively though the loss function $\ell(\cdot)$ is not differentiable everywhere. In particular, we propose a algorithm based on stochastic gradient descent (SGD) [Zhang 2004] to solve the optimization problem (12).

The pseudo-code of the proposed SGD algorithm is summarized in Algorithm 2. Specifically, in each iteration t of the algorithm, we first choose a set $A_t \subseteq \mathcal{D}_S$ of size k, and a set $B_t \subseteq \mathcal{D}_T$ of size k. Then we define a new objective function as follows,

$$\begin{aligned} &\min_{\mathbf{V},\boldsymbol{\Theta},\mathbf{W}} \widetilde{h}(\mathbf{V}, \boldsymbol{\Theta}, \mathbf{W}; A_t, B_t) \qquad (14)\\ &= \frac{1}{2kc}\sum_{(\mathbf{x}_{S_i}, y_{S_i})\in A_t}\sum_{\alpha}\ell\left((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \boldsymbol{\Theta}, \mathbf{W}\right) + \frac{\lambda_1}{2}\|\mathbf{W}\|_F^2 + \frac{\lambda_2}{2}\operatorname{tr}\left(\boldsymbol{\Theta}^\top\boldsymbol{\Upsilon}_1^t\boldsymbol{\Theta}\right) + \frac{\lambda_3}{2}\|\boldsymbol{\Theta}\|_F^2,\\ &\text{s.t. } \|\mathbf{V}_i\|_2 \leq 1, \end{aligned}$$

where $\boldsymbol{\Upsilon}_1^t$ denotes the computation of $\boldsymbol{\Upsilon}_1$ as defined in (7) on the sets A_t and B_t. Note that $\boldsymbol{\Upsilon}_1$ is dependent on the mapping $\mathbf{V}$.

ALGORITHM 2: SGD for solving $\mathbf{V}$, $\boldsymbol{\Theta}$ and $\mathbf{W}$ in (11) jointly.

```
.  Input: $\mathcal{D}_S$, $\mathcal{D}_T$ and parameters: $\lambda_1, \lambda_2, \lambda_3, T, \eta_t, p, q$ and $k$.
.  Output: $\mathbf{V}$, $\boldsymbol{\Theta}$ and $\mathbf{W}$.
1:  Initialize $\mathbf{W}_0$, $\boldsymbol{\Theta}_0$ and $\mathbf{V}_0$.
2:  for $t = 0$ to $T - 1$ do
3:      Pick random $A_t \subseteq \mathcal{D}_S$ and $B_t \subseteq \mathcal{D}_T$, such that $|A_t| = |B_t| = 2k$.
4:      Denote
        $A_t^+ = \{(\mathbf{x}_{S_i}, y_{S_i}, \alpha) : (\mathbf{x}_{S_i}, y_{S_i}) \in A_t, \alpha \in \{1, \ldots, c\}, \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}_t, \boldsymbol{\Theta}_t, \mathbf{W}_t) > 0\}$.
5:      Fix $\boldsymbol{\Theta}_t$ and $\mathbf{V}_t$, update $\mathbf{W}_{t+1}$ by using (15) and (16).
6:      Fix $\mathbf{W}_{t+1}$ and $\mathbf{V}_t$, update $\boldsymbol{\Theta}_{t+1}$ by using (17) and (18).
7:      Fix $\mathbf{W}_{t+1}$ and $\boldsymbol{\Theta}_{t+1}$, update $\mathbf{V}_{t+\frac{1}{2}}$ by using (19) and (20).
8:      Project each row of $\mathbf{V}_{t+\frac{1}{2}}$ to a unit vector by using (22).
9:  end for
10: Set $\mathbf{W} = \mathbf{W}_T$, $\boldsymbol{\Theta} = \boldsymbol{\Theta}_T$ and $\mathbf{V} = \mathbf{V}_T$.
```

4.5.1. *Updating* $\mathbf{W}$.

Firstly, we fix $\boldsymbol{\Theta}_t$ and $\mathbf{V}_t$, and update $\mathbf{W}_t$ by using the following rule,

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t\nabla_{\mathbf{W}}^{(t)}\widetilde{h}(\mathbf{V}_t, \boldsymbol{\Theta}_t, \mathbf{W}_t; A_t, B_t), \tag{15}$$

where $\nabla_{\mathbf{W}}^{(t)}\widetilde{h}(\mathbf{V}_t, \boldsymbol{\Theta}_t, \mathbf{W}_t; A_t, B_t)$ is the subgradient of $\widetilde{h}(\cdot)$ with respect to $\mathbf{W}$ at $\mathbf{W}_t$ in the t^{th} iteration, which can be derived from the following proposition. The proof can be found in Appendix A.

PROPOSITION 4.1. *Given $\boldsymbol{\Theta}_t$ and $\mathbf{V}_t$, the subgradient of $\widetilde{h}(\cdot)$ with respect to $\mathbf{W}$ at $\mathbf{W}_t$ in the t^{th} iteration can be written as*

$$\nabla_{\mathbf{W}}^{(t)}\widetilde{h}(\mathbf{V}_t, \boldsymbol{\Theta}_t, \mathbf{W}_t; A_t, B_t) = \lambda_1\mathbf{W}_t + \frac{1}{kc}\sum_{(\mathbf{x}_{S_i}, y_{S_i})\in A_t^+}\boldsymbol{\Theta}_t^\top\mathbf{x}_{S_i}^\top\boldsymbol{\Phi}_{i\alpha}\mathbf{V}_t, \tag{16}$$

where $\mathbf{\Phi}_{i\alpha} = \phi(\alpha) - \phi(y_{S_i})$, *and*

$$A_t^+ = \{(\mathbf{x}_{S_i}, y_{S_i}, \alpha) : (\mathbf{x}_{S_i}, y_{S_i}) \in A_t, \alpha \in \{1, \ldots, c\}, \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha), \mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_t) > 0\}.$$

4.5.2. Updating $\mathbf{\Theta}$. After updating $\mathbf{W}_{t+1}$, we then fix $\mathbf{V}_t$ and $\mathbf{W}_{t+1}$, and update $\mathbf{\Theta}_t$ by using the following rule,

$$\mathbf{\Theta}_{t+1} = \mathbf{\Theta}_t - \eta_t \nabla_{\mathbf{\Theta}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_{t+1}; A_t, B_t), \tag{17}$$

where $\nabla_{\mathbf{\Theta}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_{t+1}; A_t, B_t)$ is the subgradient of $\widetilde{h}$ with respect to $\mathbf{\Theta}$ at $\mathbf{\Theta}_t$ in the t^{th} iteration, which can be derived from the following proposition. The proof can be found in Appendix B.

PROPOSITION 4.2. *Given* $\mathbf{V}_t$ *and* $\mathbf{W}_{t+1}$, *the subgradient of* $\widetilde{h}(\cdot)$ *with respect to* $\mathbf{\Theta}$ *at* $\mathbf{\Theta}_t$ *in the* t^{th} *iteration can be written as*

$$\nabla_{\mathbf{\Theta}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_{t+1}; A_t, B_t) = \left(\lambda_3 \mathbf{I} + \lambda_2 \mathbf{\Upsilon}_1^t\right)\mathbf{\Theta}_t + \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} \mathbf{x}_{S_i}^\top \mathbf{\Phi}_{i\alpha} \mathbf{V}_t \mathbf{W}_{t+1}^\top. \tag{18}$$

4.5.3. Updating $\mathbf{V}$. Finally, by fixing $\mathbf{\Theta}_{t+1}$ and $\mathbf{W}_{t+1}$, we can update $\mathbf{V}_t$ by using the rules derived from the following proposition. The proof can be found in Appendix C.

PROPOSITION 4.3. *Given updated* $\mathbf{\Theta}_{t+1}$ *and* $\mathbf{W}_{t+1}$, *the mapping* $\mathbf{V}_t$ *can be updated by the following two steps,*

(*1*) *We first update* $\mathbf{V}_{t+\frac{1}{2}}$ *as follows,*

$$\mathbf{V}_{t+\frac{1}{2}} = \mathbf{V}_t - \eta_t \nabla_{\mathbf{V}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_{t+1}, \mathbf{W}_{t+1}; A_t, B_t), \tag{19}$$

where $\nabla_{\mathbf{V}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_{t+1}, \mathbf{W}_{t+1}; A_t, B_t)$ *is the subgradient of* $\widetilde{h}$ *with respect to* $\mathbf{V}$ *in the* t^{th} *iteration, which can be written as*

$$\begin{aligned} &\nabla_{\mathbf{V}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_{t+1}, \mathbf{W}_{t+1}; A_t, B_t) \\ &= -\lambda_2 \mathbf{\Upsilon}_2^t \mathbf{V}_t + \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} \left(\mathbf{\Psi}_{i\alpha} \mathbf{V}_t + \mathbf{\Phi}_{i\alpha} \mathbf{x}_{S_i} \mathbf{\Theta}_{t+1} \mathbf{W}_{t+1}\right), \end{aligned} \tag{20}$$

where $\mathbf{\Upsilon}_2$ *is defined as follows,*

$$\mathbf{\Upsilon}_2 = \frac{1}{n_1^2} \phi(\mathbf{Y}_S)^\top \mathbf{H} \mathbf{X}_S \mathbf{\Theta} \mathbf{\Theta}^\top \mathbf{X}_S^\top \mathbf{H} \phi(\mathbf{Y}_S), \tag{21}$$

and $\mathbf{\Upsilon}_2^t$ *is the computation of* $\mathbf{\Upsilon}_2$ *on the set* A_t. $\mathbf{\Psi}_{i\alpha} = \phi(y_{S_i})^\top \phi(y_{S_i}) - \phi(\alpha)^\top \phi(\alpha)$. *Note that* $\mathbf{\Upsilon}_2$ *is dependent on the mapping* $\mathbf{\Theta}$.

(*2*) *We then project each of row of* $\mathbf{V}_{t+\frac{1}{2}}$ *onto the set*

$$B = \left\{\mathbf{v} : \mathbf{v} \in \mathbb{R}^{1 \times q}, \|\mathbf{v}\|_2 \leq 1\right\}$$

by using the following equation,

$$(\mathbf{V}_{t+1})_i = \min\left(1, \frac{1}{\left\|\left(\mathbf{V}_{t+\frac{1}{2}}\right)_i\right\|_2}\right)\left(\mathbf{V}_{t+\frac{1}{2}}\right)_i. \tag{22}$$

In general, gradient-descent-based methods on non-convex functions may converge to undesirable local minima. One way to avoid this is to provide a *good* initialization or prior. Note that the solutions of $\mathbf{V}_0$, $\mathbf{\Theta}_0$ and $\mathbf{W}_0$ obtained from the sequential-optimization problems (5), (7) and (9) may be highly accurate initializations for solving the joint-optimization (11). Therefore, we can use $\mathbf{V}_0$, $\mathbf{\Theta}_0$ and $\mathbf{W}_0$ as initial start points of $\mathbf{V}$, $\mathbf{\Theta}$ and $\mathbf{W}$ for the SGD algorithm in Algorithm 2.

4.6. Further Discussion

For the optimization problem proposed in (11), we can further enforce a set of margins $\{\mathbf{C}_{y_i,\alpha} \geq 0\}$'s in the equality constraints as follows,

$$\begin{aligned} \min_{\mathbf{V},\boldsymbol{\Theta},\mathbf{W}} \ & \frac{1}{2n_1c}\sum_{i,\alpha}\xi_{i\alpha} + \frac{\lambda_1}{2}\|\mathbf{W}\|_F^2 + \frac{\lambda_2}{2}\operatorname{tr}\left(\boldsymbol{\Theta}^\top\boldsymbol{\Upsilon}_1\boldsymbol{\Theta}\right) + \frac{\lambda_3}{2}\|\boldsymbol{\Theta}\|_F^2 \\ \text{s.t.} \ & \left\|\phi(y_{S_i})\mathbf{V} - \mathbf{x}_{S_i}\boldsymbol{\Theta}\mathbf{W}\right\|_2^2 + \mathbf{C}_{y_i,\alpha} \leq \left\|\phi(\alpha)\mathbf{V} - \mathbf{x}_{S_i}\boldsymbol{\Theta}\mathbf{W}\right\|_2^2 + \xi_{i\alpha},\ \forall\alpha \in \mathcal{Y}\backslash y_{S_i}, \\ & \xi_{i\alpha} \geq 0,\ i = 1,\ldots,n_1, \\ & \|\mathbf{V}_i\|_2 \leq 1,\ \&\ \mathbf{C}_{y_i,\alpha} \geq 0, \end{aligned} \tag{23}$$

such that a specific source domain input $\mathbf{x}_i$ is much closer to its corresponding label-prototype than others by a *large margin* $\mathbf{C}_{y_i,\alpha}$. The values of the margins can be converted by a cost matrix $\mathbf{C}$ between classes. However, in general, cost matrices may not be available. For NER, there may exist a hierarchical structure underlying different types of entities including the "negative" type of entities. A simple solution to generate a cost matrix is to use the shortest paths between entities on the hierarchy as their misclassification cost. However, based on our experiments, we find that by using the shortest paths between entities as cost to define large margins $\{\mathbf{C}_{y_i,\alpha}\}$'s in (23), we do not get a better solution compared to that obtained by solving (11). The reason may be that most NER tasks are class imbalanced. The number of "negative" entities always dominates the number of "positive" entities. Even for "positive" entities, some of them are relatively rarer than others. Therefore, in order to construct a proper cost matrix between entities, besides utilizing the shortest paths on the entity hierarchy, we need to take the class-imbalanced issue into consideration as well. We leave this research issue in our future work.

4.7. Computational Complexity

In multiclass classification, one important issue is the computational cost in testing. For each testing instance (e.g., a term or word), TJE requires $O(mp)$ to project the instance from the original feature space to the feature latent space, and $O(pq)$ to apply the nearest neighbor rule (10) to make predictions, where m is the dimensionality of the original feature space, p is the dimensionality of the feature latent space, and q is the dimensionality of the label latent space. In general, $p \ll m$, and $q \ll c$, where c is the number of labels/classes. Thus the total computational time for a testing instance is $O(p(m+q))$. By applying the label embedding tree algorithm in the label latent space as proposed in Bengio et al. [2010], the testing time can be further reduced to $O(p(m+log(q)))$, which can dramatically reduce the computational time when the number of classes is huge.

5. EXPERIMENTS

5.1. Datasets

In our experiments, we use the Automatic Content Extraction (ACE) 2005 dataset,[3] which consists of 6 domains: Broadcast Conversations (BC), Broadcast News (BN), Conversational Telephone Speech (CTS), Newswire (NW), Usenet (UN), and Weblog (WL). A brief description of the 6 domains is presented in Table I. In total, there are 7 entity types and 40 subtypes annotated in the corpus. The detailed information of the types and subtypes of entities is described in Table II, where FAC, GPE, LOC, ORG, PER, VEH and WEA denote *Facility*, *Geo-Political Entity*, *Location*, *Organization*, *Person*,

[3]http://www.itl.nist.gov/iad/mig/tests/ace/ace05/doc/ace05-evalplan.v3.pdf

Table I. Description of Domains of the ACE 2005 Dataset

Domains	Sources of news articles
BC	CNN CrossFire, CNN Inside Politics, and CNN Late Edition
BN	Cable News Network, CNN Headline News
NW	Agence France Presse in English, Associated Press, New York Times, Xinhua News Agency in English
CTS	EARS Fisher 2004 Telephone Speech Collection Supplement
UN	Various internet discussion forums / bulletin boards
WL	Various internet weblogs (shared online journals)

Table II. Summary of Entity Types and Subtypes of the ACE 2005 Dataset

Types	Subtypes
FAC	Airport, Building-Grounds, Path, Plant, Subarea-Facility
GPE	Continent, County-or-District, GPE-Cluster, Nation, Population-Center, Special, State-or-Province
LOC	Address, Boundary, Celestial, Land-Region-Natural, Region-General, Region-International, Water-Body
ORG	Commercial, Educational, Entertainment, Government, Media, Medical-Science, Non-Governmental, Religious, Sports
PER	Group, Indeterminate, Individual
VEH	Air, Land, Subarea-Vehicle, Underspecified, Water
WEA	Biological, Blunt, Chemical, Exploding, Nuclear, Projectile, Sharp, Shooting, Underspecified

Table III. Summary of the ACE 2005 Dataset on NER

Domain	# Sentences	# Words	# Words per sentence	% Words of positive types	# Features	# Features after filtering
BC	2,711	45,911	16.9	9.32%	41,589	12,165
BN	3,892	62,352	16.0	6.89%		
NW	1,995	53,325	26.7	12.59%		
CTS	2,017	42,380	21.0	6.83%		
UN	1,853	41,797	22.5	6.32%		
WL	3,138	49,601	15.8	6.45%		

Vehicle, and *Weapon*, respectively. Some previous results have been reported on the 7 entity types in cross-domain NER [Jiang and Zhai 2007; Wu et al. 2009]. However, it is more useful to recognize more meaningful types of entities for real-world applications, such as airports, detailed addresses, etc. Therefore, in this article, the evaluation of different models in cross-domain NER is performed on the 40 subtypes. The 40 subtypes of entities can be referred to as *positive classes*. By considering "none-of-the-above" as an additional type of entities, which can be referred to as the *negative class*, we have 41 classes in total. We use conventional features to represent each word for NER, including lexical features such as current word and context words in a window, orthographic features that capture capitalization, digitalization and other word formation information, gazetteer features derived from the presence of the current word string in name lists, part-of-speech features, and semantic trigger word features for head noun of entity names or their local context, which are commonly in machine-learning-based NER systems [Zhou and Su 2002]. Since the features are extremely sparse, we filter out the features whose total frequencies are lower than 5 for computational efficiency. After that, we obtain a relatively small dictionary with around 12,000 features. The summary of the statistic of the dataset is described in Table III.

In this dataset, we construct 30 cross-domain NER tasks, which are listed in Table IV. In the table, the word before an arrow corresponds with the source domain and the

Table IV. The 30 Cross-Domain NER Tasks Conducted on the ACE 2005 Dataset

Source domains	Target domains					
	BC	BN	NW	CTS	UN	WL
BC	-	BC → BN	BC → NW	BC → CTS	BC → UN	BC → WL
BN	BN → BC	-	BN → NW	BN → CTS	BN → UN	BN → WL
NW	NW → BC	NW → BN	-	NW → CTS	NW → UN	NW → WL
CTS	CTS → BC	CTS → BN	CTS → NW	-	CTS → UN	CTS → WL
UN	UN → BC	UN → BN	UN → NW	UN → CTS	-	UN → WL
WL	WL → BC	WL → BN	WL → NW	WL → CTS	WL → UN	-

word after an arrow corresponds with the target domain. For each task, we randomly sample 1,000 sentences (around 19,000 words or instances) from the source domain. The words in the sampled sentences with their labels (entity types or "none-of-the-above") are considered as the labeled source domain data $\mathcal{D}_S$. We also randomly sample 800 sentences (around 15,000 words) from the target domain. The words in the sampled sentences without the labels are considered as the unlabeled target domain data $\mathcal{D}_T$. Finally, we randomly sample the other 1,000 sentences (around 19,000 words) from the target domain. The words in the sampled sentences with labels are considered as target domain test data $\mathcal{D}_T^*$. We train different models on $\mathcal{D}_S$ and $\mathcal{D}_T$, and evaluate them on $\mathcal{D}_T^*$. We repeat this process 5 times, and report the average results.

5.2. Evaluation Criteria

For evaluation criteria, we use *micro* and *macro* F1 scores to evaluate the NER results.[4] The definition of F1 for binary classes (positive or negative) is defined as follows,

$$\text{F1} = 2 \cdot \frac{\text{Pre} \cdot \text{Rec}}{\text{Pre} + \text{Rec}}, \tag{24}$$

where Pre is the precision which is the proportion of the positive instances that are correctly predicted against all positive predictions[5]

$$\text{Pre} = \frac{|\{\text{true positive instances}\} \cap \{\text{predicted positive instances}\}|}{|\{\text{predicted positive instances}\}|},$$

and Rec is the recall which is the proportion of the positive instances that are correctly predicted against all true positive instances,

$$\text{Rec} = \frac{|\{\text{true positive instances}\} \cap \{\text{predicted positive instances}\}|}{|\{\text{true positive instances}\}|}.$$

Both F1-micro and F1-macro are extensions of F1 in the multiclass setting. For NER on the ACE 2005 dataset, we have 40 entity types $\mathcal{C}_p = \{1, 2, \ldots, 40\}$ and 1 negative

[4]Note that typically NER is evaluated at phrase-level. However, for evaluation on entity subtypes on the ACE 2005 dataset, the ACE organizers suggested to use a hybrid criterion of both word-level and phrase-level. In general, the performance of NER at phrase-level can be improved if one uses the "B-I-O" tagging strategy for each subtype. However, if we use the B-I-O tags on each subtype, many tags are extremely sparse in some domains. Therefore, in this work, we evaluate NER at word level. It is still fair that we compare all methods using the same criterion.
[5]We assume the positive class is the class of interest.

type $\mathcal{C}_n = \{41\}$. Precision and recall used in F1-micro are defined as follows,

$$\text{Pre-micro} = \frac{|\{\text{instances annotated as one of the 40 entity types that are predicted correctly}\}}{|\{\text{instances predicted as one of the 40 entity types}\}|},$$

$$\text{Rec-micro} = \frac{|\{\text{instances annotated as one of the 40 entity types that are predicted correctly}\}|}{|\{\text{instances annotated as one of the 40 entity types}\}|}.$$

Based on Pre-micro and Rec-micro, F1-micro can be calculated accordingly. Differently, F1-macro is defined as follows,

$$\text{F1-macro} = \sum_{i \in \mathcal{C}_p} \frac{n_i}{N} \text{F1}_i, \tag{25}$$

where n_i is the number of instances annotated as the i^{th} entity type, N is the total number of instances annotated as one of the 40 entity types, and F1_i is the F1 defined in (24) by considering the instances annotated as the i^{th} entity type as positive instances and the rest as negative instances.

5.3. Baselines

To investigate the effectiveness of our proposed method, we compare it with the following methods, including state-of-the-art domain adaptation methods in NER.

—In-domain classifier MC_i, which is a multiclass classifier trained with labeled data from the target domain. For example, for the task BC $\rightarrow$ BN, MC_i corresponds a classifier trained with the labeled data from the BN domain. Therefore, the performance of MC_i for the task BC $\rightarrow$ BN can be also regarded as an upper bound of other cross-domain tasks whose target domain is BN, that is, NW $\rightarrow$ BN, CTS $\rightarrow$ BN, UN $\rightarrow$ BN, and WL $\rightarrow$ BN. To build an in-domain multiclass classifier, we try three methods: 1) the *one-vs-rest* (1vsR) strategy, 2) [Crammer and Singer 2002]'s multiclass SVMs,[6] and 3) hierarchical SVMs [Dumais and Chen 2000], which applies SVMs on the taxonomy of the 41 classes. The entity taxonomy is shown in Figure 1. In training hierarchical SVMs, we first train a SVM to identify whether a term is a named entity or not. And for all "positive" entities, we train a multiclass SVM to classify a term into one of the 7 entity types. Finally, for each type of entity, we train a multiclass SVM to classify a term into its corresponding entity subtypes.[7] In testing using hierarchical SVMs, along the taxonomy, which SVMs to be used is based on the predicted results of the topper layer SVMs.[8]

—No-transfer classifiers, which are trained with the source domain labeled data $\mathcal{D}_S$ and applied to make predictions on the target domain data $\mathcal{D}_T^*$ directly without adaptation. We denote $\text{MC}1_{vsR}$, MC_c, and MC_h the multiclass classifiers built with the 1vsR strategy, [Crammer and Singer 2002]'s method, and hierarchical SVMs, respectively.

—Adaptive domain bootstrapping (DAB), which is one of state-of-the-art instance-based domain adaptation methods for NER. Note that DAB was proposed in the

[6]We use LIBLINEAR SVM [Fan et al. 2008] to implement [Crammer and Singer 2002]'s multiclass SVMs, and the 1vsR strategy.

[7]In hierarchical SVMs, each SVM has its own hyper-parameters, which are tuned on some handout data.

[8]We find that the multiclass SVM with the 1vsR strategy outperforms better than the other two. Therefore, in this article, we use the multiclass SVM built with the 1vsR strategy for the in-domain classifier, and use it as the basic classifier for other transfer learning baselines as well.

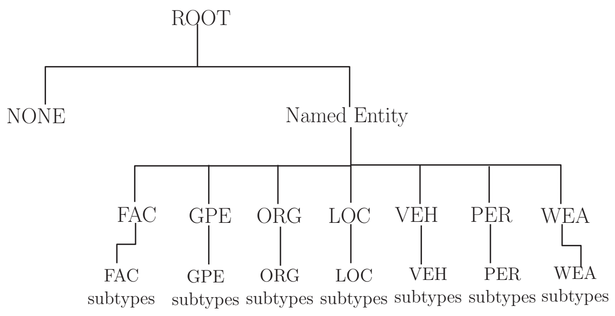


Fig. 1. Named entity hierarchy used in hierarchical SVMs.

transductive learning manner. Therefore, we conduct two implementation versions of DAB for comparison. The first version denoted by DAB$_t$ is to apply DAB on $\mathcal{D}_S$ and $\mathcal{D}_T^*$ to train a model and make predictions on $\mathcal{D}_T^*$ at the same time. The second version denoted by DAB$_o$ is to apply DAB on $\mathcal{D}_S$ and $\mathcal{D}_T$ to train a model, and use the learned model to make predictions on $\mathcal{D}_T^*$. The latter version is more practical for real-world applications, since in many scenarios, the test $\mathcal{D}_T^*$ are not available in training.

—Structural correspondence learning (SCL), which is a state-of-the-art feature-based domain adaptation methods for NLP. For comparison, we apply SCL on $\mathcal{D}_S$ and $\mathcal{D}_T$ to train a model, and apply the learned model to make predictions on $\mathcal{D}_T^*$. Following Blitzer et al. [2007], we apply mutual information on the features and labels on the source domain data to select "pivot" features. The numbers of the pivot features and new derived features are tuned on some heldout data.

—As mentioned when $\mathbf{\Theta} = \mathbf{I}$, TJE is reduced to the label embedding method [Bengio et al. 2010] for single domain multiclass classification. We denote this method LE, and consider it as another baseline method.

—SSTCA, which is another state-of-the-art feature extraction method for domain adaptation, can also be regarded as a reduction of TJE. Therefore, we consider it as a baseline for comparison as well. We first apply SSTCA on $\mathcal{D}_S$ and $\mathcal{D}_T$ to learn a new feature representation, and then train a multiclass classifier with the new representations of $\mathcal{D}_S$. For testing, we first transform $\mathcal{D}_T^*$ using the new feature presentation, and use the multiclass classifier to make predictions. For the kernel on labels in SSTCA, we use the multiclass kernel proposed by Song et al. [2007].

For DAB$_t$, DAB$_o$, SCL, and SSTCA, we use 1vsR SVMs as the base classifiers. All parameters of baselines are tuned on some heldout data on the task BC $\rightarrow$ BN and are fixed to be used for all tasks.

5.4. Implementation of TJE for NER

In this section, we describe the implementation details of our proposed method TJE for NER. As mentioned in the previous section, we can implement TJE by using two different optimization strategies. One is based on sequential optimization TJE$_q$ (Algorithm 1) and the other is based on joint optimization TJE$_j$ (Algorithm 2). The parameter settings of TJE$_q$ and TJE$_j$ are summarized in Table V. For the dimensionality q of the label embedding space, which corresponds to the mapping $\mathbf{V} \in \mathbb{R}^{c\times q}$, we set $q = c$. This is because on the ACE dataset, the total number of classes c is only 41 (including the negative type of entities). In general, when c is large, we can set $q \ll c$ to make the learning and prediction more efficient. For the learning rate η_t in each iteration t, we set $\eta_t = 0.1/t$, such than when t increases, the learning rate η_t decreases.

Table V. Parameter Setting of TJE

	λ_1	λ_2	λ_3 (μ for TJE$_q$)	T	η_t	p	q	k
TJE$_q$	10^{-4}	N/A	10^{-2}	N/A	N/A	500	41	N/A
TJE$_j$	10^{-4}	10^{-3}	10^{-2}	30	$0.1/t$	500	41	# pos. instances in $\mathcal{D}_S$

For constructing A_t and B_t in each iteration t, we set k be the number of instances annotated as any of the 40 entity subtypes in $\mathcal{D}_S$. More specifically, we first select all instances annotated as one of the 40 entity subtypes, and then randomly select k negative instances from D_S to construct A_t. After that we randomly select $2k$ instances from $\mathcal{D}_T$ to construct B_t. The reason of this instance selection strategy is that from Table III we can find that the number of words annotated as *positive* types is much smaller than the number of words annotated as the negative type. If we randomly select $2k$ instances from D_S to construct A_t, most of them are negative instances. However, the goal of NER is to extract positive entities, and further classify them into predefined categories. Our proposed instance selection strategy is to ensure in each iteration there are sufficient positive instances to be selected to update the mappings $\mathbf{\Theta}$, $\mathbf{W}$ and $\mathbf{V}$, which can be referred to as oversampling in class imbalanced classification problems [Chawla et al. 2002]. Similar to the baseline algorithms, parameters of TJE$_q$ and TJE$_j$ are tuned on some heldout data on the task BC $\rightarrow$ BN, and are fixed to be used for all tasks. The sensitivity study on parameters is reported in Section 5.7.

5.5. Comparison Results

We first compare TJE$_q$ and TJE$_j$ with MC$_i$, MC$_c$, MC$_{1vsR}$, MC$_h$, DAB$_t$, DAB$_o$, and SCL on the 30 tasks in terms of micro and macro F1 in Tables VI and VII. From the overall performance, the feature-based domain adaptation methods, such as SCL, TJE$_q$, and TJE$_j$ outperform the instance-based domain adaptation methods, such as DAB$_t$, and DAB$_o$. The reason is that for different sources of new articles, the vocabularies used can be very different, and even the syntactic structures can be very different as well (e.g., Cable News Network vs. internet weblogs), resulting in a possibly small percentage of overlapping features across different domains of news articles. In this case, reusing the source domain labeled data by reweighting cannot help boost NER performance in the target domain. In contrast, feature-based domain adaptation methods try to discover some *implicit* features or *latent factors* underlying the observations of different domains such that the distance between different domains represented by these implicit features is reduced. If there exist such implicit features, then knowledge transfer across domains based on these features becomes possible.

More specifically, consider Broadcast News (BN) as the target domain as an example. The news articles in the BN domain of the ACE 2005 dataset do not include any capitalized words, while whether a word is capitalized or not is an important discriminative feature to classify named entities in the other five domains. In this case, the weight of the word-capitalization feature in the model learned from any of the other five domains does not have any impact on making predictions on BN articles. In general, there exist other features having similar properties as the word-capitalization feature in the BN domain, which results in bad cross-domain NER performance (i.e., the second group of the results in Tables VI and VII) of the no-transfer methods (i.e., MC$_c$, MC$_{1vsR}$, MC$_h$), and the instance-based domain adaptation methods (i.e., DAB$_t$, and DAB$_o$). In contrast, the feature-based domain adaptation methods (i.e., SCL, TJE$_q$, and TJE$_j$) aim to learn implicit features across domains, and then train models onto the new feature representations. If there exist discriminative implicit features across the source and target domains, it is still possible to achieve good cross-domain NER performance even though the explicit features across domains are very different. Furthermore, compared

Table VI. Comparison Results on Cross-Domain 40-Entity-Type NER (F1-micro, unit:%)

Src.	Tar.	F1-micro								
		MC_i	MC_c	MC_{1vsR}	MC_h	DAB_t	DAB_o	SCL	TJE_q	TJE_j
BN			75.70	77.77	77.12	67.98	64.60	**78.85**	76.65	77.36
NW			75.00	76.05	73.66	**78.05**	77.97	77.24	77.67	77.91
CTS	BC	*84.83*	70.25	70.45	69.49	74.34	73.07	73.36	73.63	**74.18**
UN			77.05	76.48	76.73	78.01	**78.56**	76.95	75.47	76.05
WL			74.77	75.86	73.20	77.35	**77.55**	77.31	76.75	77.53
BC			54.97	54.91	44.05	49.75	55.97	64.34	67.91	**68.21**
NW			58.67	59.10	40.13	53.94	55.28	**66.08**	65.07	65.19
CTS	BN	*78.70*	35.84	38.18	20.10	25.22	21.52	55.63	**70.28**	70.17
UN			63.71	65.87	57.45	67.60	67.61	69.22	**74.45**	74.17
WL			51.40	56.40	41.17	53.16	53.41	56.58	70.68	**71.86**
BC			71.45	72.56	71.03	71.97	71.29	72.04	72.11	**72.47**
BN			72.39	74.70	73.77	60.99	62.40	72.60	74.42	**74.78**
CTS	NW	*85.15*	69.40	70.20	68.67	73.93	**75.71**	68.86	74.59	74.80
UN			74.27	75.40	74.91	76.31	**76.96**	76.57	76.49	76.42
WL			76.98	77.35	75.40	81.08	**81.24**	79.07	77.84	80.22
BC			85.09	85.50	85.03	85.26	86.06	**87.27**	86.69	86.71
BN			82.29	84.29	83.71	61.06	44.53	**87.49**	86.62	86.89
NW	CTS	*94.13*	87.48	88.04	83.34	88.15	88.34	**89.06**	84.64	84.29
UN			87.90	87.90	87.55	88.40	88.25	**89.14**	87.60	87.78
WL			83.39	83.66	82.77	83.97	84.31	86.55	86.66	**87.01**
BC			65.80	67.85	64.86	66.30	65.17	66.27	66.64	**68.54**
BN			67.20	69.30	68.48	61.28	63.30	67.34	70.84	**71.92**
NW	UN	*81.73*	71.53	72.55	67.99	72.01	72.83	**73.79**	73.33	73.27
CTS			66.32	68.36	65.83	65.91	68.36	67.77	69.24	**70.09**
WL			72.00	72.97	68.39	74.05	74.62	74.00	73.34	**76.18**
BC			57.14	**59.18**	57.25	55.69	56.56	57.79	57.32	58.68
BN			62.30	63.44	61.99	45.20	49.17	62.95	63.66	**64.17**
NW	WL	*76.56*	66.95	68.07	65.27	66.84	67.02	66.86	68.19	**68.74**
CTS			56.65	57.47	55.27	63.65	64.44	55.43	63.97	**64.70**
UN			63.85	64.21	62.84	**69.28**	69.21	64.03	66.35	66.44
Avg.		*83.44*	69.26	70.47	66.58	67.89	67.84	72.02	73.64	**74.23**

to SCL, besides discovering an adapted feature latent space, TJE_q and TJE_j learn a latent space for labels. This label latent space can capture the relationships between labels, which is useful when the number of labels is large, especially when the "positive" training instances are sparse. Therefore, for cross-domain NER on the 40-type entities, both TJE_q and TJE_j perform better than SCL. Finally, benefit from the joint optimization strategy, TJE_j performs slightly better than TJE_q. Note that besides the difference on optimization strategies, TJE_q uses the least-squared loss while TJE_j uses the hinge loss. In our experiments, we have verified that by replacing the least-squared loss with the hinge loss in TJE_q, TJE_j still performs slightly better that TJE_q.

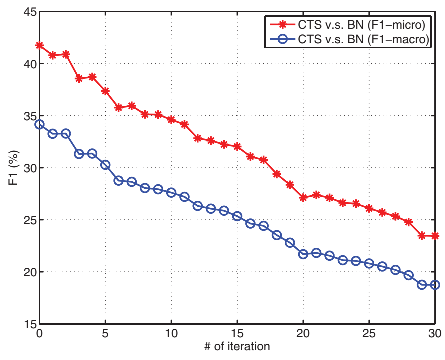
Furthermore, consider Conversational Telephone Speech (CTS) as the target domain as another example. From the fourth group of the results in Tables VI and VII, we can observe that the no-transfer methods perform well in these five cross-domain NER tasks, which implies that most discriminative features of the CTS domain appear frequently and behave similarly on the other 5 domains. In this case, the feature-based domain adaptation methods do not achieve much improvement compared to the no-transfer learning methods and the instance-based domain adaptation methods. In

Table VII. Comparison Results on Cross-Domain 40-Entity-Type NER (F1-macro, unit:%)

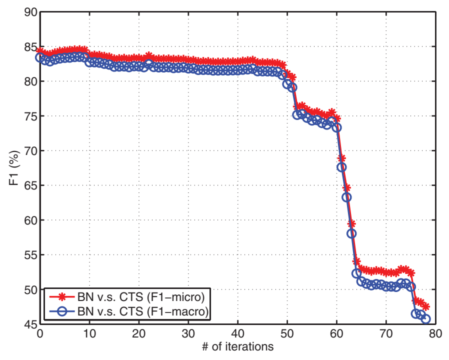
Source domain	Target domain	F1-macro								
		MC_i	MC_c	MC_{1vsR}	MC_h	DAB_t	DAB_o	SCL	TJE_q	TJE_j
BN			73.41	75.83	75.40	62.68	58.88	**76.12**	74.71	75.56
NW			74.58	75.00	73.69	76.53	**76.59**	75.61	75.37	75.68
CTS	BC	*83.05*	67.23	67.03	67.08	70.08	68.31	67.76	69.84	**70.50**
UN			75.01	74.26	75.14	75.12	**75.99**	73.70	72.76	74.06
WL			73.91	74.00	73.16	75.66	75.91	74.68	75.06	**76.02**
BC			50.30	49.62	38.89	42.68	49.59	58.76	62.78	**63.17**
NW			53.94	54.57	36.64	46.00	47.12	**60.63**	60.14	60.29
CTS	BN	*77.21*	29.75	30.38	15.60	19.13	16.58	48.25	**64.97**	64.58
UN			57.77	61.37	52.05	61.44	61.88	63.14	**71.58**	71.36
WL			47.08	52.34	37.60	47.88	48.03	50.62	67.81	**68.98**
BC			69.27	**69.84**	68.78	67.95	67.38	68.04	69.03	69.11
BN			70.72	**73.19**	72.39	54.60	56.22	69.78	72.66	73.07
CTS	NW	*84.01*	65.12	65.30	64.69	69.38	71.60	62.38	71.09	**71.38**
UN			71.73	72.54	72.34	73.08	73.98	72.27	73.83	**74.23**
WL			75.22	75.46	74.07	79.38	**79.78**	76.41	76.36	78.79
BC			85.15	85.24	84.94	84.69	85.60	**85.84**	85.00	85.07
BN			81.47	83.38	82.95	58.86	42.45	**86.23**	85.79	86.03
NW	CTS	*93.35*	87.59	87.72	86.52	87.74	87.94	**88.23**	83.67	83.32
UN			86.92	86.77	86.64	87.22	87.13	**87.82**	86.38	86.73
WL			82.63	83.04	82.68	82.94	83.29	85.52	85.54	**85.97**
BC			64.43	**66.00**	63.77	62.69	61.67	62.76	63.61	65.94
BN			66.44	68.15	67.35	58.18	59.32	65.33	69.02	**70.18**
NW	UN	*80.36*	70.75	71.32	67.55	69.24	70.21	**71.44**	70.45	70.44
CTS			64.05	65.54	63.85	61.52	64.60	63.08	65.83	**67.09**
WL			71.00	71.14	68.13	72.19	72.80	71.35	71.32	**74.22**
BC			53.00	**54.27**	52.84	48.54	50.26	51.35	51.93	53.80
BN			60.16	61.03	59.28	39.01	44.80	59.98	61.49	**61.67**
NW	WL	*74.54*	65.57	**66.42**	63.84	64.02	64.74	64.02	65.26	65.95
CTS			50.28	50.43	49.81	57.12	59.32	47.25	60.65	**61.47**
UN			60.97	60.46	59.14	**65.87**	65.74	58.62	63.73	63.93
Avg.		*82.09*	66.85	67.72	63.81	64.03	64.26	68.23	70.92	**71.62**

addition, different from TJE_q and TJE_j, SCL augments the original features with the learned implicit features. If most original features are useful for multiple domains, feature augmentation can further boost the performance, resulting in that SCL performs the best in this group of experiments.

Another interesting observation is that DAB_t and DAB_o perform abnormally on some tasks, such as the tasks BN $\rightarrow$ BC, CTS $\rightarrow$ BN, BN $\rightarrow$ NW, BN $\rightarrow$ CTS and BN $\rightarrow$ WL. The performance of DAB_t and DAB_o on these tasks is even much worse than that of the no-transfer methods. The reason is that DAB is a bootstrapping-style-based method, whose performance highly depends on quality of the selected instances in each iterations, especially in first few iterations, and the termination criterion. If the quality of the selected instances are "bad" (i.e., the instances that are very dissimilar to the current training pool, and thus cannot be classified correctly), DAB may suffer from error propagation during iterations. Specifically, consider the task CTS $\rightarrow$ BN as an example. As shown in Figure 2(a), by adding "bad" instances from the target domain in the first few iterations, the performance of DAB_o in terms of micro and macro F1 scores decreases dramatically when the number of iterations increases. Another example is



(a) Varying # of iterations of DAB$_o$ on CTS → BN.



(b) Varying # of iterations of DAB$_o$ on BN → CTS.

Fig. 2. Performance of DAB$_o$ in terms of micro and macro F1 scores under varying numbers of iterations.

Table VIII. Impact of Different Components of TJE on 40-Entity-Type NER (unit: %)

Target domain	F1-micro				F1-macro			
	SSTCA	LE	TJE$_q$	TJE$_j$	SSTCA	LE	TJE$_q$	TJE$_j$
BC (avg.)	75.32	**76.55**	76.03	76.61	73.29	73.04	73.55	**74.36**
BN (avg.)	55.48	64.26	69.68	**69.92**	50.32	58.70	65.46	**65.68**
NW (avg.)	72.32	74.61	75.09	**75.74**	69.03	70.99	72.60	**73.32**
CTS (avg.)	84.68	**86.87**	86.44	86.54	83.74	**85.46**	85.28	85.42
UN (avg.)	68.49	69.93	70.68	**72.00**	67.23	66.21	68.04	**69.57**
WL (avg.)	61.82	62.72	63.90	**64.55**	57.59	58.44	60.61	**61.37**
Avg. (all)	69.69	72.42	73.64	**74.23**	66.87	68.81	70.92	**71.62**

presented in Figure 2(a), which shows the performance of DAB$_o$ in terms of micro and macro F1 scores on the task BN → CTS when the number of iterations increases. We can observe that in the 10th iteration, DAB$_o$ achieves its best performance on the task. However, DAB$_o$ fails to discover an appropriate termination condition. By adding more and more bootstrapped target domain data, the classification performance of DAB$_o$ in terms of micro and macro F1 scores finally drops from around 84% to around 45%. In contrast, similar to other feature-based transfer learning approaches, our proposed TJE$_q$ and TJE$_j$ aim to discover an optimal adaptive feature representation in one step, which can avoid the bad-instance selection and termination issues, and thus perform more stably on the 30 tasks.

5.6. Impact of Different Components of TJE

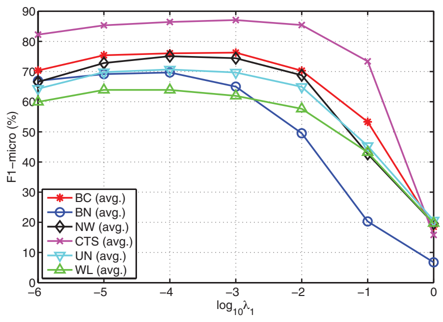
As introduced in Section 4, our proposed TJE consists of two main components: 1) embedding the labels into a latent space to capture their intrinsic relationships, and 2) discovering an intermediate feature latent space underlying the source and target domain data to reduce the domain difference. Therefore, we conduct experiments to test the impact of each component to the overall performance of TJE. If we do not embed the discrete labels into the latent space, then TJE is approximately reduced to SSTCA. In addition, as mentioned above, if we set $\mathbf{\Theta} = \mathbf{I}$, which implies to drop the intermediate feature learning process, TJE is reduced to LE. The comparison results among TJE, SSTCA and LE on the 30 tasks are summarized in Table VIII. In the table, each row corresponds to the average results of different models on the five tasks with the same target domain. For example, BC (avg.) denotes the average results of the tasks BN → BC, NW → BC, CTS → BC, UN → BC and WL → BC. As we can see from the table, by only learning a powerful feature representation across domains, SSTCA cannot benefit much for cross-domain NER. This is because though SSTCA can find an feature space to reduce domain difference, it may suffer from the label imbalanced issue by decomposing the multiclass NER task into multiple binary classification tasks. The performance of LE shows that exploiting the relationship between classes in the latent space is important for NER, especially when the number of positive entities is much smaller than that of the negative ones. TJE integrates these two learning processes into a unified framework and thus achieves the best performance in cross-domain NER.

5.7. Parameter Sensitivity Study of TJE

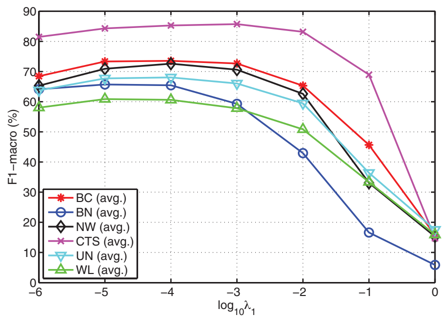
In the third experiment, we study the effect of different parameter settings to the overall performance of TJE_j in terms of micro and macro F1 scores. When we test the sensitivity of one parameter, we fix the values of the other parameters. In TJE_j, λ_1, λ_2 and λ_3 are tradeoff parameters to balance the impact of different objective terms. Figure 3 shows the overall performance of TJE_j in terms of micro and macro F1 scores under varying values of λ_1, λ_2 and λ_3, respectively. From the figure, we can observe that when $\lambda_1 \leq 10^{-3}$, $\lambda_2 \leq 10^{1}$ and $\lambda_3 \leq 10^{-1}$, TJE_j performs well and stably for most cross-domain tasks. In Figures 4(a) and 4(b), we show the effect of different values of the dimensionality p, which corresponds to the transfer feature mapping $\mathbf{\Theta} \in \mathbb{R}^{m\times p}$, to the overall performance of TJE_j in terms of micro and macro F1 scores. We can find that for most tasks, when p is in the range of [300 600], TJE_j performs well and stably. In Figures 4(c) and 4(d), we show the effect of different values of the dimensionality q, which corresponds to the label mapping $\mathbf{V} \in \mathbb{R}^{c\times q}$, to the overall performance of TJE_j in terms of micro and macro F1 scores. We can observe that for most tasks, when $q \geq 30$, TJE_j performs well and stably. Finally, we test the convergence property of TJE_j. In Figures 4(e) and 4(f), we show the average performance of TJE_j in terms of micro and macro F1 scores on the 30 tasks when the number of iterations increases from 1 to 30. As we can see, TJE_j quickly converges to good performance in terms of micro and macro F1 scores when $T \geq 20$.

5.8. Further Empirical Analysis

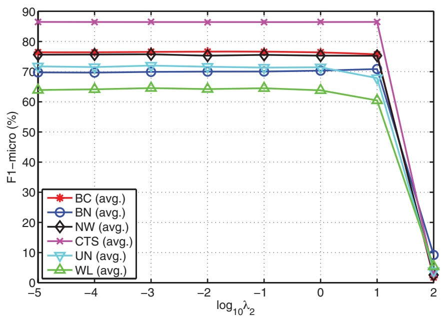
In this section, in order to further understand why we propose to embed labels into a latent space for cross-domain NER, we conduct cross-domain NER experiments on the 7 types of entities of the ACE 2005 dataset (as described in Table II). The comparison results are presented in Tables IX and X. As we can see that though the number of classes is reduced from 41 to 8 (including the negative type of entities), our proposed TJE_j still performs best on average of the 30 tasks. However, the improvement of TJE_j compared to SCL become much smaller than that presented in Tables VI and VII. SCL even performs better than TJE_q slightly. The reason is that for each type of entities which contains several subtypes of entities, the number of instances is much larger than that of a subtype. Though the cross-domain NER tasks still suffer from the class imbalanced issue, the ratio between the numbers of entities of a positive type and the negative type become much larger. Therefore, the benefit obtained from the label latent space become less than that obtained from the embedding of the 41 classes. This is similar to the reason that the improvement of LE compared to MC_c and MC_{1vsR} on the 8 classes of entities is much smaller than that on the 41 classes of entities. The comparison observations between Tables IX–X and Tables VI–VII suggest that when the number of entity types gets larger, our proposed method is supposed to get better performance compared to the other baselines. Note that the results of DAB_t and DAB_o presented in Tables VI and VII are different from the previous results reported in Wu et al. [2009]. The reason are two folds 1) the feature sets used to represent each term or word are different. In this work, the features we use are the same as those used in



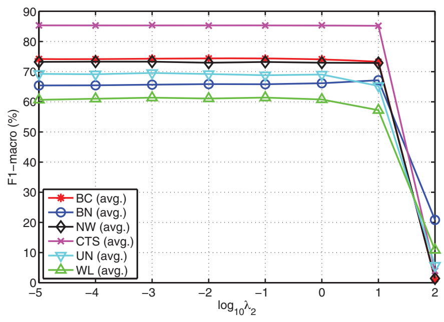
(a) Varying values of parameter λ_1 w.r.t. F1-micro.



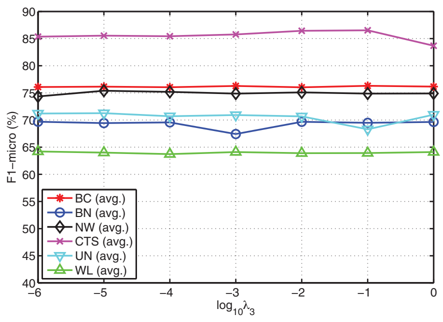
(b) Varying values of parameter λ_1 w.r.t. F1-macro.



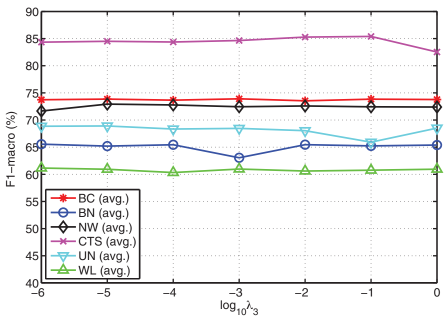
(c) Varying values of parameter λ_2 w.r.t. F1-micro.



(d) Varying values of parameter λ_2 w.r.t. F1-macro.



(e) Varying values of parameter λ_3 w.r.t. F1-micro.



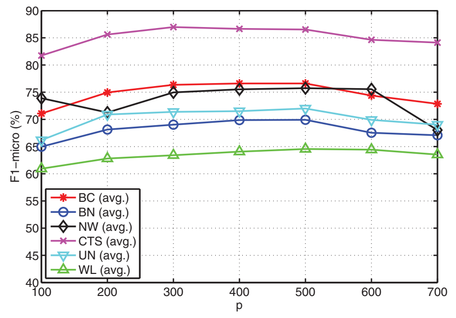
(f) Varying values of parameter λ_3 w.r.t. F1-macro.

Fig. 3. Performance of TJE$_j$ in terms of micro and macro F1 scores under varying values of parameters λ_1, λ_2 and λ_3.

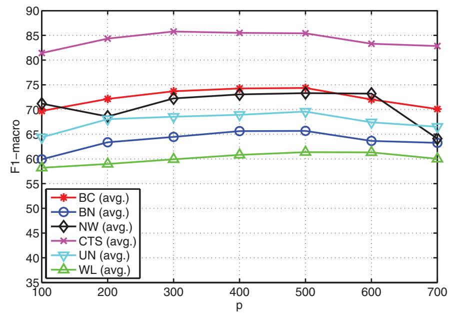
our current NER system, which are state-of-the-art features for NER combining with some features based on our feature engineering experience, and 2) in this work, we report all the 30 cross-domain NER tasks of the ACE 2005 dataset, while in Wu et al. [2009], only 12 of them are reported.

6. CONCLUSION AND FUTURE WORK

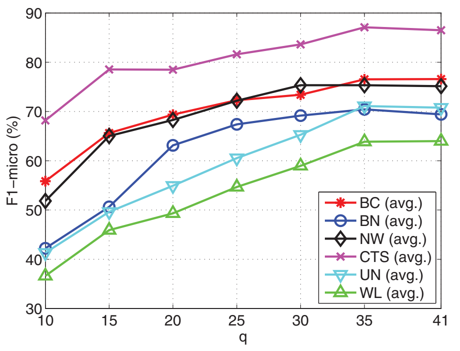
In this article, we propose a new transfer learning framework, named Transfer Joint Embedding (TJE), for cross-domain multiclass classification with application to NER. The motivation of TJE is to project both labels and features onto a same



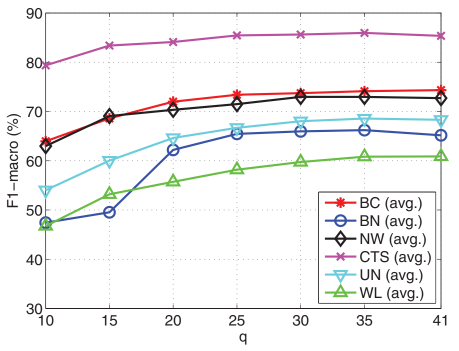
(a) Varying # of dim. p w.r.t. F1-micro.



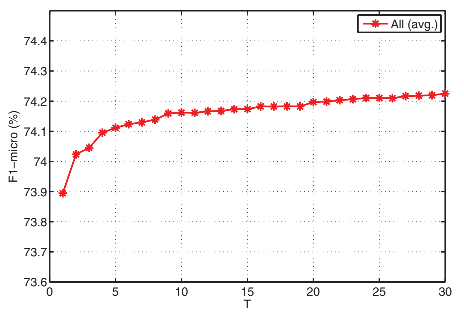
(b) Varying # of dim. p w.r.t. F1-macro.



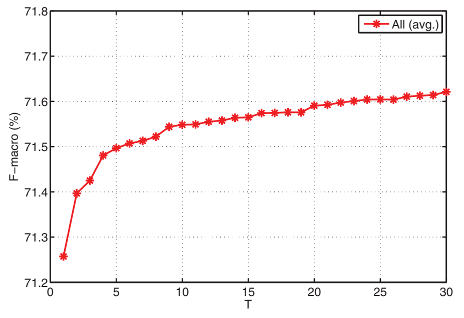
(c) Varying # of dim. q w.r.t. F1-micro.



(d) Varying # of dim. q w.r.t. F1-macro.



(e) Varying # of iterations T w.r.t. F1-micro.



(f) Varying # of iterations T w.r.t. F1-macro.

Fig. 4. Performance of TJE$_j$ under varying numbers of dimensionality p, q, and iterations T.

low-dimensional latent space, where the relationships between labels can be fully exploited, the distance in data distributions between the source and target domains can be reduced, and the projected source domain instances are closer to their corresponding label-prototypes than others. In this way, classification on the target domain test data can be done with the simple nearest neighbor rule. We propose two solutions for TJE by using different optimization strategies. One denoted by TJE$_q$ is based on sequential optimization on different variables, and the other denoted by TJE$_j$ is based on joint optimization on all variables. Experimental results on the ACE 2005 dataset demonstrate the effectiveness of TJE.

Table IX. All Comparison Results on Cross-Domain 7-Entity-Type NER (F1-micro, unit:%)

Target domain	F1-micro										
	MC_i	MC_c	MC_{1vsR}	MC_h	DAB_t	DAB_o	SCL	SSTCA	LE	TJE_q	TJE_j
BC (avg.)	*85.14*	77.34	77.97	76.42	75.69	75.55	**79.25**	77.39	77.70	77.73	77.81
BN (avg.)	*78.96*	52.39	53.96	40.10	43.73	45.55	62.13	50.53	63.18	67.13	**68.88**
NW (avg.)	*87.97*	77.68	**79.29**	78.42	73.57	74.57	79.26	77.41	77.45	77.83	77.89
CTS (avg.)	*95.03*	86.60	87.31	86.95	84.96	86.25	**89.15**	85.75	88.16	88.41	88.59
UN (avg.)	*83.03*	72.17	72.32	71.31	69.80	68.84	73.25	69.79	72.66	73.44	**73.88**
WL (avg.)	*78.92*	66.66	67.10	66.34	60.66	60.00	**67.71**	66.85	65.53	65.85	65.71
Avg. (all)	*84.84*	72.14	72.98	69.92	68.07	68.46	75.13	71.29	74.11	75.06	**75.76**

Table X. All Comparison Results on Cross-Domain 7-Entity-Type NER (F1-Macro, unit:%)

Target domain	F1-macro										
	MC_i	MC_c	MC_{1vsR}	MC_h	DAB_t	DAB_o	SCL	SSTCA	LE	TJE_q	TJE_j
BC (avg.)	*85.14*	76.17	76.77	75.77	73.29	73.15	**77.71**	76.05	75.50	76.07	76.32
BN (avg.)	*78.96*	49.73	51.29	37.28	40.07	42.37	58.01	48.26	58.83	63.95	**65.81**
NW (avg.)	*87.97*	76.03	**77.52**	**77.00**	69.83	71.50	76.76	75.58	74.90	75.58	76.17
CTS (avg.)	*95.03*	86.69	87.41	87.73	84.76	86.17	**89.91**	85.53	87.80	88.16	88.38
UN (avg.)	*83.03*	70.93	71.12	70.54	67.68	66.69	71.90	68.78	70.68	71.74	**72.32**
WL (avg.)	*78.92*	64.89	64.98	64.89	56.66	55.91	65.42	64.94	63.24	63.96	65.43
Avg. (all)	*84.18*	70.74	71.51	68.87	65.38	65.96	73.96	69.85	71.83	73.25	**74.07**

In the future, as discussed in Section 4.6, we will study the problem on how to encode the hierarchical structure of the types of entities into learning the label latent space to further boost the cross-domain NER performance. Furthermore, we plan to apply TJE to other datasets which contains large-scale types of named entities to further verify the effectiveness of TJE. Besides cross-domain NER, we also plan to apply TJE to solve other cross-domain natural language processing tasks, such as cross-domain relation extraction and cross-domain opinion extraction.

APPENDIX

A. PROOF OF PROPOSITION 4.1

PROOF. By fixing $\mathbf{\Theta}$ and $\mathbf{V}$, the optimization problem (14) can be written as the following unconstrained form by dropping terms which are independent to $\mathbf{W}$,

$$\min_{\mathbf{W}} \widetilde{h}(\mathbf{V}, \mathbf{\Theta}, \mathbf{W}; A_t, B_t) = \frac{1}{2kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t} \sum_{\alpha} \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \mathbf{\Theta}, \mathbf{W}) + \frac{\lambda_1}{2} \|\mathbf{W}\|_F^2. \quad (26)$$

Denote

$$A_t^+ = \{(\mathbf{x}_{S_i}, y_{S_i}, \alpha) : (\mathbf{x}_{S_i}, y_{S_i}) \in A_t, \alpha \in \{1, \ldots, c\}, \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_t) > 0\}. \quad (27)$$

Note that $A_t^+ \subseteq A_t$. The loss function $\ell(.)$ is not differentiable with respect to $\mathbf{W}$ on A_t but differentiable on A_t^+. Furthermore, $\ell(.) > 0$ on A_t^+ while $\ell(.) = 0$ on $A_t \backslash A_t^+$. It can be proved that the subgradient of the first term on the right-hand side in (26) with respect to $\mathbf{W}$ on A_t in the t^{th} iteration can be written as

$$\nabla_{\mathbf{W}}^{(t)} \frac{1}{2kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t} \sum_{\alpha} \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \mathbf{\Theta}, \mathbf{W}) = \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} \mathbf{\Theta}_t^\top \mathbf{x}_{S_i}^\top \mathbf{\Phi}_{i\alpha} \mathbf{V}_t,$$

where $\mathbf{\Phi}_{i\alpha} = \phi(\alpha) - \phi(y_{S_i})$. Therefore, the subgradient of $\widetilde{h}(\cdot)$ with respect to $\mathbf{W}$ on A_t and B_t in the t^{th} iteration can be written as

$$\nabla_{\mathbf{W}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_t; A_t, B_t) = \lambda_1 \mathbf{W}_t + \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} \mathbf{\Theta}_t^\top \mathbf{x}_{S_i}^\top \mathbf{\Phi}_{i\alpha} \mathbf{V}_t.$$

The proof of the proposition is completed. □

B. PROOF OF PROPOSITION 4.2

PROOF. By fixing $\mathbf{V}$ and $\mathbf{W}$, the optimization problem (14) can be written as the following unconstrained form by dropping terms which are independent to $\mathbf{\Theta}$,

$$\min_{\mathbf{\Theta}} \widetilde{h}(\mathbf{V}, \mathbf{\Theta}, \mathbf{W}; A_t, B_t) \tag{28}$$

$$= \frac{1}{2kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t} \sum_{\alpha} \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \mathbf{\Theta}, \mathbf{W}) + \frac{\lambda_2}{2} \operatorname{tr}\left(\mathbf{\Theta}^\top \mathbf{\Upsilon}_1^t \mathbf{\Theta}\right) + \frac{\lambda_3}{2} \|\mathbf{\Theta}\|_F^2. \tag{29}$$

Note that the loss function $\ell(.)$ is not differentiable with respect to $\mathbf{\Theta}$ on A_t but differentiable on A_t^+, which is defined in (27), and $\ell(.) > 0$ on A_t^+ while $\ell(.) = 0$ on $A_t \backslash A_t^+$. It can be proved that the subgradient of the first term on the right-hand side in (28) with respect to $\mathbf{\Theta}$ on A_t in the t^{th} iteration can be written as

$$\nabla_{\mathbf{\Theta}}^{(t)} \left(\frac{1}{2kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t} \sum_{\alpha} \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \mathbf{\Theta}, \mathbf{W}) \right) = \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} \mathbf{x}_{S_i}^\top \mathbf{\Phi}_{i\alpha} \mathbf{V}_t \mathbf{W}_{t+1}^\top.$$

Therefore, the subgradient of $\widetilde{h}(\cdot)$ with respect to $\mathbf{\Theta}$ on A_t and B_t in the t^{th} iteration can be written as

$$\nabla_{\mathbf{\Theta}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_t, \mathbf{W}_{t+1}; A_t, B_t) = \lambda_3 \mathbf{\Theta}_t + \lambda_2 \mathbf{\Upsilon}_1^t \mathbf{\Theta}_t + \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} \mathbf{x}_{S_i}^\top \mathbf{\Phi}_{i\alpha} \mathbf{V}_t \mathbf{W}_{t+1}^\top.$$

This completes the proof of the proposition. □

C. PROOF OF PROPOSITION 4.3

PROOF. By fixing $\mathbf{W}$ and $\mathbf{\Theta}$, the optimization problem (14) can be written as the following inequality-constrained minimization problem by dropping terms which are independent to $\mathbf{V}$,

$$\min_{\mathbf{V}} \widetilde{h}(\mathbf{V}, \mathbf{\Theta}, \mathbf{W}; A_t, B_t) = \frac{1}{2kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t} \sum_{\alpha} \ell((\mathbf{x}_{S_i}, y_{S_i}, \alpha); \mathbf{V}, \mathbf{\Theta}, \mathbf{W}) + \frac{\lambda_2}{2} \operatorname{tr}\left(\mathbf{\Theta}^\top \mathbf{\Upsilon}_1^t \mathbf{\Theta}\right),$$

$$\text{s.t.} \quad \|\mathbf{V}_i\|_2 \leq 1. \tag{30}$$

For solving this problem, we can use subgradient projection methods. More specifically, we can first update $\mathbf{V}$ by using subgradient decent, and then project it to the set defined by constraints. Similar to updating $\mathbf{W}$ and $\mathbf{\Theta}$, it can be shown that the subgradient of $\widetilde{h}(\cdot)$ with respect to $\mathbf{V}$ on A_t and B_t in the t^{th} iteration can be written as

$$\nabla_{\mathbf{V}}^{(t)} \widetilde{h}(\mathbf{V}_t, \mathbf{\Theta}_{t+1}, \mathbf{W}_{t+1}; A_t, B_t) = -\lambda_2 \mathbf{\Upsilon}_2^t \mathbf{V}_t + \frac{1}{kc} \sum_{(\mathbf{x}_{S_i}, y_{S_i}) \in A_t^+} (\mathbf{\Psi}_{i\alpha} \mathbf{V}_t + \mathbf{\Phi}_{i\alpha} \mathbf{x}_{S_i} \mathbf{\Theta}_{t+1} \mathbf{W}_{t+1}),$$

where $\boldsymbol{\Upsilon}_2^t$ is defined in (21). Denote $\mathbf{V}_{t+\frac{1}{2}}$ the updated values of $\mathbf{V}$ after applying subgradient descent on the t^{th} iteration,

$$\mathbf{V}_{t+\frac{1}{2}} = \mathbf{V}_t - \eta_t \nabla_{\mathbf{V}}^{(t)} \widetilde{h}(\mathbf{V}_t, \boldsymbol{\Theta}_{t+1}, \mathbf{W}_{t+1}; A_t, B_t).$$

In order to satisfy the constraints $\|\mathbf{V}_i\|_2 \leq 1, i \in \{1, \ldots, c\}$, we can post-project each row of $\mathbf{V}_{t+\frac{1}{2}}$ to the unit ball $B = \{\mathbf{v} : \mathbf{v} \in \mathbb{R}^{1\times q}, \|\mathbf{v}\|_2 \leq 1\}$ by using the following equation,

$$(\mathbf{V}_{t+1})_i = \min\left(1, \frac{1}{\left\|\left(\mathbf{V}_{t+\frac{1}{2}}\right)_i\right\|_2}\right)\left(\mathbf{V}_{t+\frac{1}{2}}\right)_i.$$

This completes the proof of the proposition. □

REFERENCES

AONE, C., HALVERSON, L., HAMPTON, T., AND RAMOS-SANTACRUZ, M. 1998. SRA: Description of the IE2 system used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*.

BELKIN, M. AND NIYOGI, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput. 15*, 6, 1373–1396.

BEN-DAVID, S., BLITZER, J., CRAMMER, K., AND PEREIRA, F. 2007. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, vol. 19, MIT Press, Cambridge, MA, 137–144.

BENGIO, S., WESTON, J., AND GRANGIER, D. 2010. Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems*, vol. 23, 163–171.

BLITZER, J., DREDZE, M., AND PEREIRA, F. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. ACL, 432–439.

BLITZER, J., MCDONALD, R., AND PEREIRA, F. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language*. 120–128.

CHAWLA, N. V., BOWYER, K. W., HALL, L. O., AND KEGELMEYER, W. P. 2002. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Resear. 16*, 321–357.

CHEN, B., LAM, W., TSANG, I. W., AND WONG, T.-L. 2009. Extracting discriminative concepts for domain adaptation in text mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 79–188.

CIARAMITA, M. AND ALTUN, Y. 2005. Named-entity recognition in novel domains with external lexical knowledge. In *Proceedings of the NIPS Workshop on Advances in Strucured Learning for Text and Speech Processing*.

COX, T. AND COX, M. 1994. *Multidimensional Scaling*. Chapman & Hall, London.

CRAMMER, K. AND SINGER, Y. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Resear. 2*, 265–292.

DAUMÉ III, H. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. ACL, 256–263.

DUMAIS, S. AND CHEN, H. 2000. Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 256–263.

FAN, R.-E., CHANG, K.-W., HSIEH, C.-J., WANG, X.-R., AND LIN, C.-J. 2008. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Resear. 9*, 1871–1874.

FINKEL, J. R., GRENAGER, T., AND MANNING, C. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. ACL, 363–370.

FINKEL, J. R. AND MANNING, C. D. 2009. Nested named entity recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. ACL, 141–150.

GLOROT, X., BORDES, A., AND BENGIO, Y. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*. 513–520.

GRETTON, A., BORGWARDT, K. M., RASCH, M., SCHÖLKOPF, B., AND SMOLA, A. 2007. A kernel method for the two-sample problem. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, 513–520.

GRETTON, A., BOUSQUET, O., SMOLA, A. J., AND SCHÖLKOPF, B. 2005. Measuring statistical dependence with Hilbert-Schmidt norms. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*.

HUMPHREYS, K., GAIZAUSKAS, R., AZZAM, S., HUYCK, C., MITCHELL, B., CUNNINGHAM, H., AND WILKS, Y. 1998. Description of the University of Sheffield LaSIE-II system as used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*.

ISOZAKI, H. AND KAZAWA, H. 2002. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th International Conference on Computational Linguistics*. ACL, 1–7.

JIANG, J. AND ZHAI, C. 2006. Exploiting domain structure for named entity recognition. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. ACL, 74–81.

JIANG, J. AND ZHAI, C. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. ACL, 264–271.

KRUPKA, G. R. AND HAUSMAN, K. 1998. Isoquest inc.: Description of the NetOwlTM extractor system as used for MUC-7. In *Proceedings of 7th Message Understanding Conference.*

KWOK, C., ETZIONI, O., AND WELD, D. S. 2001. Scaling question answering to the web. *ACM Trans. Inf. Syst. 19*, 242–262.

MANNING, C. D., RAGHAVAN, P., AND SCHTZE, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY.

MIKHEEV, A., GROVER, C., AND MOENS, M. 1998. Description of the LTG system used for MUC-7. In *Proceedings of the 7th Message Understanding Conference*.

MIKHEEV, A., GROVER, C., AND MOENS, M. 1999. Named entity recognition without gazetteers. In *Proceedings of the 19th International Conference of the European Chapter of the Association for Computational Linguistics*. 1–8.

NADEAU, D. AND SEKINE, S. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes 30*, 1, 3–26.

PAN, S. J., KWOK, J. T., AND YANG, Q. 2008. Transfer learning via dimensionality reduction. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. 677–682.

PAN, S. J., NI, X., SUN, J.-T., YANG, Q., AND CHEN, Z. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 751–760.

PAN, S. J., TSANG, I. W., KWOK, J. T., AND YANG, Q. 2011. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks 22*, 199–210.

PAN, S. J. AND YANG, Q. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng. 22*, 10, 1345–1359.

QUIONERO-CANDELA, J., SUGIYAMA, M., SCHWAIGHOFER, A., AND LAWRENCE, N. D. 2009. *Dataset Shift in Machine Learning*. MIT Press.

RABINER, L. R. AND JUANG, B. H. 1986. An introduction to hidden Markov models. *IEEE ASSP Mag. 3*, 1, 4–16.

SCHUMAKER, R. P. AND CHEN, H. 2009. Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Trans. Inf. Syst. 27*, 12:1–12:19.

SEKINE, S., SUDO, K., AND NOBATA, C. 2002. Extended named entity hierarchy. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation.* 1818–1824.

SMOLA, A. J., GRETTON, A., SONG, L., AND SCHÖLKOPF, B. 2007. A Hilbert space embedding for distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*. 13–31.

SONG, L. 2007. Learning via Hilbert space embedding of distributions. Ph.D. thesis, University of Sydney.

WEINBERGER, K. Q. AND CHAPELLE, O. 2009. Large margin taxonomy embedding for document categorization. In *Advances in Neural Information Processing Systems*, vol. 21, 1737–1744.

WHITELAW, C., KEHLENBECK, A., PETROVIC, N., AND UNGAR, L. 2008. Web-scale named entity recognition. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, 123–132.

WU, D., LEE, W. S., YE, N., AND CHIEU, H. L. 2009. Domain adaptive bootstrapping for named entity recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. ACL, 1523–1532.

ZHANG, T. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning*. ACM, 116–123.

ZHOU, G. AND SU, J. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. ACL, 473–480.

Received October 2011; revised July 2012, November 2012; accepted January 2013