Cross-domain Sentiment Classification using Sentiment Sensitive Embeddings

Danushka Bollegala *Member, IEEE*, Tingting Mu *Member, IEEE*, John Y. Goulermas *Senior Member, IEEE*

Abstract—Unsupervised Cross-domain Sentiment Classification is the task of adapting a sentiment classifier trained on a particular domain (*source domain*), to a different domain (*target domain*), without requiring any labeled data for the target domain. By adapting an existing sentiment classifier to previously unseen target domains, we can avoid the cost for manual data annotation for the target domain. We model this problem as embedding learning, and construct three objective functions that capture: (a) distributional properties of *pivots* (i.e. common features that appear in both source and target domains), (b) label constraints in the source domain documents, and, (c) geometric properties in the unlabeled documents in both source and target domains. Unlike prior proposals that first learn a lower-dimensional embedding independent of the source domain sentiment labels, and next a sentiment classifier in this embedding, our joint optimisation method learns embeddings that are sensitive to sentiment classification. Experimental results on a benchmark dataset show that by jointly optimising the three objectives we can obtain better performances in comparison to optimising each objective function separately, thereby demonstrating the importance of task-specific embedding learning for cross-domain sentiment classification. Among the individual objective functions, the best performance is obtained by (c). Moreover, the proposed method reports cross-domain sentiment classification accuracies that are statistically comparable to the current state-of-the-art embedding learning methods for cross-domain sentiment classification.

Keywords—Domain Adaptation, Sentiment Classification, Spectral Methods, Embedding Learning

1 Introduction

The ability to correctly identify the sentiment expressed in user-reviews about a particular product is an important task for several reasons. First, if there is a negative sentiment associated with a particular feature of a product, the manufacturer can take immediate actions to address the issue. Failing to detect a negative sentiment associated with a product might result in decreased sales. From the users' point-of-view, in online stores where one cannot physically touch and evaluate a product as in a real-world store, the user opinions are the only available subjective descriptors of the product. By automatically classifying the user-reviews according to the sentiment expressed in them, we can assist the potential buyers of a product to easily understand the overall opinion about that product. Considering the numerous applications of sentiment classification such as opinion mining [1], opinion summarisation [2], contextual advertising [3], and market analysis [4], it is not surprising that sentiment classification has received continuous attention.

Sentiment classification can be considered as an instance of text classification where a given *document* must be classified into a pre-defined set of sentiment classes [5]. We use the term *document* to refer various types of user reviews. In binary sentiment classification, a document must be classified into two classes depending on whether it expresses a *positive* or a *negative* sentiment towards an entity. Alternatively, a document can be assigned a discrete sentiment score (eg. from one to five stars) that indicates the degree of the positivity (or negativity) of the sentiment. Once, a document has been identified as sentiment bearing, then further analysis can be performed, for example, to extract evidence for an argument.

In supervised binary sentiment classification, a binary classifier is trained using manually labeled positive and negative user-reviews. Considering the vast number of products sold online, it is both costly as well as infeasible to manually annotate reviews for each product type. On the other hand, it is attractive if we could some how *adapt* a sentiment classifier that is trained using labeled reviews for one product to classify sentiment on a different product. This problem setting is known as *Cross-Domain Sentiment Classification*. For example, consider the situation where we have trained a sentiment classifier using labeled reviews for *books* and would like to apply it to classify sentiment on kitchen utensils such as *knives*. We use the term *domain* to refer to a collection of reviews written on a particular product. The domain from which we train our sentiment classifier is called the *source*, whereas the domain to which we apply the trained classifier is called the *target*. In our example, *books* is the source domain and *knives* is the target domain. Words such as *interesting*, *exciting*, or *boring* are used to express sentiment about books in reviews, whereas words such as *durable*, *sharp*, or *lightweight* are used

• *The authors are with the School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, Brownlow Hill, Liverpool, L69 3BX, United Kingdom.*
E-mail: {danushka.bollegala, t.mu, j.y.goulermas}@liverpool.ac.uk

to express sentiment about knives. Unfortunately, this mismatch of features between the source and target domains causes a sentiment classifier trained on *books* to perform poorly when applied to *knives*.

Domain adaptation methods can be further classified into two groups: *supervised domain adaptation methods* [6]–[8], and *unsupervised domain adaptation methods* [9]–[12]. In supervised domain adaptation, one assumes the availability of a small labeled dataset for the target domain in addition to the labeled data for the source domain, and unlabeled data for both the source and the target domains. On the other hand, unsupervised domain adaptation does not assume the availability of labeled data for the target domain. Although supervised domain adaptation methods often outperform unsupervised ones, the extra burden to annotate labeled data for each of the target domains is a concern. For example, in large-scale online shopping sites such as the Amazon.com [13], we must adapt to a large number of novel target domains. Consequently, in this paper, we consider **unsupervised cross-domain sentiment classification**.

One popular solution to cross-domain sentiment classification is to first project the source and the target features into the same lower-dimensional embedding, and subsequently learn a sentiment classifier on this embedded feature space [9], [11]. This approach is particularly attractive when there is little overlap between the original source and the target feature spaces. Similarly distributed words in the source and the target domains get mapped to closer points in the embedded space, thereby reducing the mismatch of features in the two domains. Prior work on cross-domain sentiment classification use unlabeled data from the source and the target domains to first learn a low-dimensional embedding for the two domains [9], [11]. Next, labeled reviews in the source domain are projected onto this embedding. Finally, a binary sentiment classifier is trained using the projected source domain labeled training instances. A limitation of this two-step approach that decouples the embedding learning and sentiment classifier training is that the embeddings learnt in the first step is agnostic to the sentiment of the documents, which is the ultimate goal in cross-domain sentiment classification.

We propose an unlabeled cross-domain sentiment classification method using spectral embeddings where we project both the words and the documents into the same lower-dimensional embedding. The embedding learnt by our method enforces three important requirements. First, a set of domain independent features (also known as *pivots*) are selected from the source and target domains which must be mapped as close as possible in the embedded space. Second, friend closeness and enemy dispersion of the source domain labeled documents must be preserved. In other words, positively labeled documents must be embedded closer to each other and far from the negatively labeled documents. Likewise, negatively labeled documents must be embedded closer to each other and far from the positively labeled documents. Third, within each domain, the local geometry among the documents must be preserved. For example, unlabeled neighbour documents in the source domain must be embedded closer to each other in the embedded space whereas, unlabeled neighbour documents in the target domain must be embedded closer to each other in the embedded space. Here, neighbour documents refer to similar documents in terms of their text content. We model each of the above-mentioned requirements as an objective function, and *jointly* optimise all three objective functions. The proposed method can be easily extended to more than two sentiment classes.

Our experimental results on a benchmark dataset for multi-domain sentiment classification demonstrate that by jointly optimising the three objectives in many cases we obtain better classification accuracies than if we had optimised each objective separately. Even in cases where joint optimisation does not improve over the separately trained objectives, the performance obtained by the joint optimisation method is never below that obtained by the best individually trained methods. This result shows the importance of learning embeddings that are sensitive to the final task at hand, which is sentiment classification. Moreover, the proposed method significantly outperforms several baselines and previously proposed embedding learning methods when applied to cross-domain sentiment classification.

2 Motivating example

To explain the motivation behind our proposed method consider the reviews shown in Table 1, where we have shown positive (indicated by $+$) and negative (indicated by $-$) labeled reviews for the *books* (source) domain and the *kitchen appliances* (target) domain. The sentiment bearing words such as *excellent*, *sharp*, *boring*, etc. are indicated by boldface. From Table 1, we see that the words that express sentiment in the two domains are very different. For example, in the source domain, words such as *interesting* and *thrilling* express a positive sentiment, whereas in the target domain the same is expressed by words such as *sharp*. This makes the task of domain adaptation a challenging one because a sentiment classifier trained on the source domain reviews is likely to perform poorly on the target domain because the features it has learnt from the source domain might not appear in the target domain. However, we also see that some words such as *excellent* appear in the reviews for both domains. Such features that represent the same sentiment in both source and target domains are referred to as *pivots* in the literature [10]. Ideally, when we embed two representations for a pivot created from the two domains, we must ensure that the projected points are embedded closer to each other. The first rule in our proposed method captures this intuition.

Next, consider the labeled reviews in the source domain. The first two reviews contain positive sentiment bearing words such as *excellent*, *interesting*, and *thrilling*.

TABLE 1: Positive (+) and negative (-) sentiment reviews in two different domains: *books* and *kitchen appliances*.

	books	*kitchen appliances*
+	This is an **excellent** survey on deep learning methods.	10 dollars for a **sharp** knife like this, could not ask for a better **bargain**.
+	The story is **interesting** and **thrilling**. Could not put the book down.	The **excellent** quality of these **sharp** knives are well **worth** their price.
-	A **disappointing** end to a **boring** story. Utter **waste** of time.	Knives got **rusty** and **blunt** after normal usage for two weeks.

Therefore, we must ensure that these two reviews are embedded to nearby points. On the other hand, the negatively labeled review in the source domain contains words such as *disappointing*, *boring*, and *waste*. Therefore, when we embed this negative labeled review we must ensure that it is embedded to a point that is distant to the embedded points corresponding to the positive labeled reviews. This requirement can be seen as a friend attraction and enemy dispersion task. The second rule we propose captures this intuition.

The distributions of words in the source domain is different from that of the target domain. Aspects of books such as the plot, length, style of writing etc. are different from that of knives such as the weight, durability, sharpness, etc. We attempt to preserve the similarities among documents from a particular domain as much as possible in the embedded space. For example, two documents that are similar in their word distributions must be embedded close to each other in the embedded space as well. Note that to measure the similarity between two documents we do not need their sentiment labels. Therefore, we can enforce this requirement for unlabeled documents in the source and the target domains. The third rule we propose captures this intuition.

The three requirements we discussed above are not mutually independent. Therefore, it is appropriate to jointly optimise the three objectives instead of optimising each one separately.

3 Related Work

Cross-domain sentiment classification methods can be classified as *unsupervised* vs. *supervised* methods. In unsupervised cross-domain sentiment classification, the training data consist of (a) source domain labeled documents, (b) source domain unlabeled documents, and (c) target domain unlabeled documents. Supervised (or semi-supervised) cross-domain sentiment classification methods use a small set of labeled data for the target domain in addition to those three data sources. Unsupervised cross-domain sentiment classification can be considered as a much harder problem because of the lack of availability of labeled data for the target domain. Unsupervised domain adaptation methods assume that the output labels in the target domain are equally conditioned by the input, even though the input could be differently distributed in terms of marginal probability. Therefore, domain adaptation methods adjust for the differences in this conditional distributions between the two domains.

Structural correspondence learning (SCL) [10] first selects a set of pivots, common features to both source and the target domains, using some criteria. One approach for selecting pivots is to select all features that occur more than a pre-defined number of times in both domains. Alternatively, a word association measure such as the mutual information (MI) could be used to measure the degree of association of a feature to a domain name, and select common features that have a high degree of association between both the source and the target domains [10]. The latter approach has shown to produce better results in cross-domain sentiment classification. Next, linear predictors are trained to predict the presence (or absence) of pivots in a document. Specifically, documents in which a particular pivot w occurs are considered as positive training instances for learning a predictor for w, whereas an equal number of documents in which w does not occur are selected as negative training instances. Unigram and bigram lexical-features are extracted from the selected training instances as features to train a binary logistic regression classifier with l_2 regularisation. Finally, the weight vector learnt by the classifier is considered as the predictor for w. The predictors learnt for all pivots are arranged in a matrix on which singular value decomposition (SVD) is performed. The left singular vectors corresponding to the largest singular values are selected from the SVD result, and arranged as row vectors in a matrix. All source domain labeled training instances are multiplied by this matrix to predict the presence of pivots. Finally, a binary logistic regression model is trained using the predicted pivots and the original features. By first predicting the pivots, and then learning a classifier using those predicted pivots as additional features, SCL attempts to reduce the mismatch between features in the source and the target domains.

Spectral feature alignment (SFA) [11] splits the feature space into two mutually exclusive groups: domain independent features (pivots), and domain specific features (all other features). Next, a bipartite graph is constructed between the two groups where the edge connecting a domain specific and a domain independent feature is weighted by the number of different documents in which the corresponding two features co-occur. Spectral clustering is performed on this bipartite graph to create a

lower dimensional representation in which co-occurring domain specific and domain independent features are represented by the same set of lower dimensional features. Similarly to SCL, SFA trains a binary logistic regression model in this lower-dimensional space using the labeled documents from the source domain. Both SFA and SCL are similar to our proposed method in that first, a lower-dimensional feature representation is learnt, and second a binary sentiment classifier is trained on this embedded space. However, our proposed method is different from SCL and SFA in that, we consider not only the unlabeled data but also labeled data for the source domain when constructing the representation. As we later see in Section 6, this enables us to learn customised representations that result in better performance on our final task of cross-domain sentiment classification.

Bollegala et al. [14] created a Sentiment Sensitive Thesaurus (SST) that lists words that express similar sentiments in the source and target domains. For example, SST created from the two domains *books* and *knives* lists *interesting* as a related word for *sharp*. The thesaurus is automatically created using a sentiment sensitive asymmetric similarity measure that uses sentiment labels in the source domain documents. Analogous to the thesauri-based query expansion in information retrieval, SST is used to expand the source domain feature vectors by appending related features in the target domain. A binary logistic regression classifier is trained using the expanded feature vectors corresponding to the source domain labeled documents. Unlike, SCL or SFA, SST does not create lower-dimensional embeddings.

Glorot et al. [13] used stacked denoising autoencoders to learn a mapping between source and target domain feature spaces. Autoencoders are pre-trained using unlabeled data from both domains, and the post-training stage fits a binary classifier on top of the trained neural network. Bollegala [15] proposed an unsupervised method to predict the distribution of a word across domains. Specifically, source and target domain feature representations are created for a set of pivots, and a multivariate regression model is trained using Partial Least Squares Regression (PLSR). They evaluate their method on cross-domain sentiment classification. However, this method does not consider source domain labeled data during the PLSR step, which makes the projection agnostic to sentiment classification. In transfer learning [16], we predict labels in the target domain different from that in the source domain, whereas in domain adaptation, we predict the same labels when the data distributions in the two domains are different.

Unsupervised dimensionality reduction methods such as the Latent Semantic Indexing (LSI) [17], Principle Component Analysis (PCA) [18], and Locality Preserving Projections (LLP) [19]–[21], generate lower-dimensional embeddings of unlabeled data points. Supervised dimensionality reduction methods such as the Fischer Discriminant Analysis (FDA) [22] considers the within and cross-class scattering of data points, when creating lower-dimensional embeddings. However, supervised dimensionality reduction methods are prone to overfitting when the number of labeled instances are small [23]. Semi-supervised dimensionality reduction methods [24], [25] are proposed to overcome this problem by taking into consideration the unlabeled data points. Our proposed method can also be seen as an instance of semi-supervised dimensionality reduction. Embedding learning is an active research field and we believe unsupervised cross-domain sentiment classification can benefit from this line of work in the future.

4 Cross-Domain Sentiment Classification

Without loss of generality, we denote the source and the target domains by A and B in the subsequent discussion. As we describe later in Section 5, different methods have been proposed for selecting a set of pivots from a given pair of domains. Our proposed embedding learning method is independent of the pivot selection step and we assume that M pivots to be given. In our experiments, the used pivot selection method is based on the pointwise mutual information between a pivot and a domain label so that words that are closely associated with both the source and the target domains are selected.

We denote the pivots by $\{U_i\}_{i=1}^{M}$. The pivots are common features to both source and the target domains and appear in both domains. However, the word context around the same pivot under different domains cannot be the same, thus, different feature vectors have to be used to characterise the same pivot in different domains. For example, for the ith pivot word, we represent it as a d-dimensional feature vector $\boldsymbol{u}_i^{(A)} = \left[u_{i1}^{(A)}, u_{i2}^{(A)}, \ldots, u_{id}^{(A)}\right]^T$ in domain A, while an h-dimensional feature vector $\boldsymbol{u}_i^{(B)} = \left[u_{i1}^{(B)}, u_{i2}^{(B)}, \ldots, u_{ih}^{(B)}\right]^T$ in domain B. These result in a vector set $\left\{\boldsymbol{u}_i^{(A)}\right\}_{i=1}^{M}$ in domain A and a set $\left\{\boldsymbol{u}_i^{(B)}\right\}_{i=1}^{M}$ in domain B, which correspond to one $M \times d$ and one $M \times h$ pivot feature matrices $\mathbf{U}_A = \left[u_{ij}^{(A)}\right]$ and $\mathbf{U}_B = \left[u_{ij}^{(B)}\right]$. In our experiment, unigrams and bigrams from the documents with which a pivot U_i co-occurs are extracted as features to obtain $\left\{\boldsymbol{u}_i^{(A)}\right\}_{i=1}^{M}$ and $\left\{\boldsymbol{u}_i^{(B)}\right\}_{i=1}^{M}$.

The remaining words are non-pivot ones, which only appear in one of the two domains. Within the same domain, we assume the documents are constructed following the same word distributions, thus, non-pivot words and the pivots are represented by the same unigram and bigram features. Letting $\{A_i\}_{i=1}^{M_A}$ denote a total of M_A non-pivot words in domain A, each is represented by a d-dimensional vector $\boldsymbol{a}_i = [a_{i1}, a_{i2}, \ldots, a_{id}]^T$, leading to the vector set $\{\boldsymbol{a}_i\}_{i=1}^{M_A}$ and the corresponding $M_A \times d$ feature matrix $\mathbf{A} = [a_{ij}]$. Similarly, we use $\{B_i\}_{i=1}^{M_B}$ to denote the

M_B non-pivot words in domain B, each represented by an$h$-dimensional vector $\boldsymbol{b}_i = [b_{i1}, b_{i2}, \ldots, b_{ih}]^T$, leading to the vector set $\{\boldsymbol{b}_i\}_{i=1}^{M_B}$ and the corresponding $M_B \times h$ feature matrix $\mathbf{B} = [b_{ij}]$.

Given a document, we characterise it by both the pivots and domain-specific non-pivot words. Let $\left\{D_i^{(A)}\right\}_{i=1}^{N_A}$ to denote a total of N_A document in domain A while $\left\{D_i^{(B)}\right\}_{i=1}^{N_B}$ the N_B documents in domain B. The document $D_i^{(A)}$ is modelled as a $(M + M_A)$-dimensional vector with its first M elements corresponding to the frequencies (alternatively some salience scores such as the tf-idf values) of the pivot words appearing in it and the next M_A elements to the frequencies (or scores) of the non-pivot words. Similarly, a document $D_i^{(B)}$ in domain B is modelled as a $(M + M_B)$-dimensional vector with its elements corresponding to the occurrence frequencies (or scores) of the M pivots and M_B non-pivot words in this document. We use the $N_A \times (M + M_A)$ and $N_B \times (M + M_B)$ feature matrices $\mathbf{X}_A = \left[x_{ij}^{(A)}\right]$, and $\mathbf{X}_B = \left[x_{ij}^{(B)}\right]$ to store feature vectors of the documents in the two domains, where ij-th element of each matrix ($x_{ij}^{(A)}$ and $x_{ij}^{(B)}$) indicates the frequency (or score) of the j-th word appearing in the i-th document.

Among the N_A documents in domain A, let us assume that there are N_{lA} labeled and N_{uA} unlabeled ones, where $N_A = N_{lA} + N_{uA}$. Different superscripts/subscripts are used to distinguish the two groups ("lA" for labeled and "uA" for unlabeled ones), for example, the $N_{lA} \times (M + M_A)$, and $N_{uA} \times (M + M_A)$ feature matrices $\mathbf{X}_{lA}$, and $\mathbf{X}_{uA}$, where $\mathbf{X}_A^T = \left[\mathbf{X}_{lA}^T, \mathbf{X}_{uA}^T\right]$. Class information of the labeled documents is stored in an $N_{lA} \times c$ binary matrix $\mathbf{Y} = [y_{ij}]$, where $y_{ij} = 1$ indicates that the i-th document belongs to the j-th class, and $y_{ij} = 0$ otherwise. Note that in binary sentiment classification we have only two classes: positive class and the negative class.

To further explore the relationships between the documents from the two domains, the distributions of the pivots under different domains, also how the word distribution affects the document distribution in the two domains, we propose to generate a k-dimensional embedding space $\mathbb{R}^k$ to map both the words $\{U_i\}_{i=1}^{M}$, $\{A_i\}_{i=1}^{M_A}$ and $\{B_i\}_{i=1}^{M_B}$ and documents $\{D_i^{(A)}\}_{i=1}^{N_A}$ and $\{D_i^{(B)}\}_{i=1}^{N_B}$ in the same space according to the following three rules:

- *Rule 1*: The same pivot $\boldsymbol{u}_i^{(A)}$ and $\boldsymbol{u}_i^{(B)}$ should be mapped as close as possible in $\mathbb{R}^k$. This preserves the word-based connection between the source and the target domains.
- *Rule 2*: The friend closeness and enemy dispersion [26] of the labeled documents in domain A should be enhanced in $\mathbb{R}^k$. This improves the class separability of documents in domain A.
- *Rule 3*: Within each domain, local geometry between documents, characterised by $\mathbf{X}_A$ (or $\mathbf{X}_B$), should be preserved in $\mathbb{R}^k$. This captures the inherent data structure within the source and target domains, and prevents the generation of an overfitted space to the small number of labeled documents.

The constructed embedding space preserves the local connections between documents built upon the pivots and non-pivot words within each domain (rule 3), meanwhile it aligns the source and target domains by matching the distributions of the pivots (rule 1). This potentially enables the two structured spaces of source and target to be connected together by the pivot words. As a result, the enhancement of the class separation in the source domain (rule 2) will be propagated to the target domain. This thus achieves unsupervised sentiment classification in the target domain through the use of supervised information in source domain enabled by the cross-domain alignment driven by the pivot information.

4.1 Mapping Function

The main strategy of mapping the words and documents to the space is to first compute the word embeddings, and then derive the document embeddings based on the word embeddings by considering the word occurrences.

Linear projection is assumed to transform the original feature representation of words to their embedding presentation. Specifically, a $d \times k$ projection matrix $\mathbf{P}_A$ is used to map words in domain A to a k-dimensional embedding space $\mathbb{R}^k$, while a $d \times h$ projection matrix $\mathbf{P}_B$ is used to map words in domain B to the same embedding space. Given in total $M + M_A$ words in domain A including the M pivots appearing in both domains and M_A non-pivot words only appearing in domain A, we let $\left\{\widetilde{\boldsymbol{z}}_i^{(A)}\right\}_{i=1}^{M+M_A}$ denote their corresponding word embeddings stored in an $(M+M_A) \times k$ embedding matrix $\widetilde{\mathbf{Z}}_A$ computed by the linear projection mapping given as

$$\widetilde{\mathbf{Z}}_A^T = \left[\mathbf{P}_A^T \mathbf{U}_A^T,\ \mathbf{P}_A^T \mathbf{A}^T\right]. \tag{1}$$

Similarly, $\left\{\widetilde{\boldsymbol{z}}_i^{(B)}\right\}_{i=1}^{M+M_B}$ denotes the embeddings for words in domain B, which results in an $(M + M_B) \times k$ embedding matrix $\widetilde{\mathbf{Z}}_B$ computed by

$$\widetilde{\mathbf{Z}}_B^T = \left[\mathbf{P}_B^T \mathbf{U}_B^T,\ \mathbf{P}_B^T \mathbf{B}^T\right]. \tag{2}$$

Here, the pivots appear in both domains, thus possess two sets of feature representations $\mathbf{U}_A$ and $\mathbf{U}_B$. Subsequently, they possess two sets of embedding representations after being mapped from the two domains, which are $\mathbf{U}_A\mathbf{P}_A$ and $\mathbf{U}_B\mathbf{P}_B$. Later on, we will show that according to rule 1 these two representations should be as similar as possible in order to reach the alignment between the two domains via pivots matching.

After computing the word embeddings $\left\{\widetilde{\boldsymbol{z}}_i^{(A)}\right\}_{i=1}^{M+M_A}$ and $\left\{\widetilde{\boldsymbol{z}}_i^{(B)}\right\}_{i=1}^{M+M_B}$, it is straightforward to derive the

TABLE 2: Summary of the main variables and quantities used in the proposed algorithm.

Variable name	Range	Description
$\{U_i\}_{i=1}^{M}$	NA	A set of M selected pivots appearing in both domains A and B.
$\{A_i\}_{i=1}^{M_A}$	NA	A set of M_A non-pivot words only appearing in domain A.
$\{B_i\}_{i=1}^{M_B}$	NA	A set of M_B non-pivot words only appearing in domain B.
$\left\{\boldsymbol{u}_i^{(A)}\right\}_{i=1}^{M}$	$\mathbb{R}^d$	A set of d-dimensional feature vectors to characterise the M pivots in domain A, $\boldsymbol{u}_i^{(A)} = \left[u_{i1}^{(A)}, u_{i2}^{(A)}, \ldots, u_{id}^{(A)}\right]^T$.
$\{\boldsymbol{a}_i\}_{i=1}^{M_A}$	$\mathbb{R}^d$	A set of d-dimensional feature vectors to characterise the M_A non-pivot words in domain A, $\boldsymbol{a}_i = [a_{i1}, a_{i2}, \ldots, a_{id}]^T$.
$\left\{\boldsymbol{u}_i^{(B)}\right\}_{i=1}^{M}$	$\mathbb{R}^h$	A set of h-dimensional feature vectors to characterise the M pivots in domain B, $\boldsymbol{u}_i^{(B)} = \left[u_{i1}^{(B)}, u_{i2}^{(B)}, \ldots, u_{ih}^{(B)}\right]^T$.
$\{\boldsymbol{b}_i\}_{i=1}^{M_B}$	$\mathbb{R}^h$	A set of h-dimensional feature vectors to characterise the M_B non-pivot words in domain B, $\boldsymbol{b}_i = [b_{i1}, b_{i2}, \ldots, b_{ih}]^T$.
$\mathbf{U}_A$	$\mathbb{R}^{M \times d}$	The pivot feature matrix in domain A, $\mathbf{U}_A = \left[u_{ij}^{(A)}\right]$.
$\mathbf{A}$	$\mathbb{R}^{M_A \times d}$	The non-pivot feature matrix in domain A, $\mathbf{A} = [a_{ij}]$.
$\mathbf{U}_B$	$\mathbb{R}^{M \times h}$	The pivot feature matrix in domain B, $\mathbf{U}_B = \left[u_{ij}^{(B)}\right]$.
$\mathbf{B}$	$\mathbb{R}^{M_B \times h}$	The non-pivot feature matrix in domain B, $\mathbf{B} = [b_{ij}]$.
$\left\{D_i^{(A)}\right\}_{i=1}^{N_A}$	NA	A set of N_A documents from domain A.
$\left\{D_i^{(B)}\right\}_{i=1}^{N_B}$	NA	A set of N_B documents from domain B.
$\mathbf{X}_{lA}$	$\mathbb{R}^{N_{lA} \times (M+M_A)}$	The feature matrix of labeled documents in domain A.
$\mathbf{X}_{uA}$	$\mathbb{R}^{N_{uA} \times (M+M_A)}$	The feature matrix of unlabeled documents in domain A.
$\mathbf{X}_A$	$\mathbb{R}^{N_A \times (M+M_A)}$	The feature matrix of all the documents in domain A, $\mathbf{X}_A^T = \left[\mathbf{X}_{lA}^T, \mathbf{X}_{uA}^T\right]$.
$\mathbf{X}_B$	$\mathbb{R}^{N_B \times (M+M_B)}$	The feature matrix of all the documents in domain B.
$\mathbf{Y}$	$\mathbb{R}^{N_{lA} \times c}$	The binary class matrix for the labeled documents in domain A.
$\left\{\widetilde{\boldsymbol{z}}_i^{(A)}\right\}_{i=1}^{M+M_A}$	$\mathbb{R}^k$	The set of k-dimensional embeddings for words in domain A.
$\left\{\widetilde{\boldsymbol{z}}_i^{(B)}\right\}_{i=1}^{M+M_B}$	$\mathbb{R}^k$	The set of k-dimensional embeddings for words in domain B.
$\mathbf{P}_A$	$\mathbb{R}^{d \times k}$	The projection matrix to map the d-dimensional words in domain A to the k-dimensional embedding space.
$\mathbf{P}_B$	$\mathbb{R}^{h \times k}$	The projection matrix to map the h-dimensional words in domain B to the k-dimensional embedding space.
$\widetilde{\mathbf{Z}}_A$	$\mathbb{R}^{(M+M_A) \times k}$	The embedding matrix of words in domain A, $\widetilde{\mathbf{Z}}_A^T = \left[\mathbf{P}_A^T \mathbf{U}_A^T, \mathbf{P}_A^T \mathbf{A}^T\right]$.
$\widetilde{\mathbf{Z}}_B$	$\mathbb{R}^{(M+M_B) \times k}$	The embedding matrix of words in domain B, $\widetilde{\mathbf{Z}}_B^T = \left[\mathbf{P}_B^T \mathbf{U}_B^T, \mathbf{P}_B^T \mathbf{B}^T\right]$.
$\left\{\boldsymbol{z}_i^{(A)}\right\}_{i=1}^{N_A}$	$\mathbb{R}^k$	The set of k-dimensional embeddings for documents in domain A.
$\left\{\boldsymbol{z}_i^{(lA)}\right\}_{i=1}^{N_{lA}}$	$\mathbb{R}^k$	The set of k-dimensional embeddings for the labeled documents in domain A.
$\left\{\boldsymbol{z}_i^{(uA)}\right\}_{i=1}^{N_{uA}}$	$\mathbb{R}^k$	The set of k-dimensional embeddings for the unlabeled documents in domain A.
$\left\{\boldsymbol{z}_i^{(B)}\right\}_{i=1}^{N_B}$	$\mathbb{R}^k$	The set of k-dimensional embeddings for documents in domain B.
$\mathbf{Z}_A$	$\mathbb{R}^{N_A \times k}$	The embedding matrix of documents in domain A, $\mathbf{Z}_A = \mathbf{D}^{-1}(\mathbf{X}_A)\mathbf{X}_A\widetilde{\mathbf{Z}}_A$.
$\mathbf{Z}_{lA}$	$\mathbb{R}^{N_{lA} \times k}$	The embedding matrix of the labeled documents in domain A.
$\mathbf{Z}_{uA}$	$\mathbb{R}^{N_{uA} \times k}$	The embedding matrix of the unlabeled documents in domain A.
$\mathbf{Z}_B$	$\mathbb{R}^{N_B \times k}$	The embedding matrix of documents in domain B, $\mathbf{Z}_B = \mathbf{D}^{-1}(\mathbf{X}_B)\mathbf{X}_B\widetilde{\mathbf{Z}}_B$.
$\mathbf{S}_{lA}$	$\mathbb{R}^{N_{lA} \times N_{lA}}$	The similarity matrix between the labeled documents in domain A, computed based on $\mathbf{X}_{lA}$.
$\mathbf{S}_{uA}$	$\mathbb{R}^{N_{uA} \times N_{uA}}$	The similarity matrix between the unlabeled documents in domain A, computed based on $\mathbf{X}_{uA}$.
$\mathbf{S}_B$	$\mathbb{R}^{N_B \times N_B}$	The similarity matrix between all the documents in domain B, computed based on $\mathbf{X}_B$.

embedding representations for documents. For example, it is possible to view the embedding vector of each document as a weighted sum of the embedding vectors of all the words that appear in the corresponding document. The occurrence frequencies (or scores) can be used as the summation weights to determine the contribution level of the words. Letting $\left\{\boldsymbol{z}_i^{(A)}\right\}_{i=1}^{N_A}$ and $\left\{\boldsymbol{z}_i^{(B)}\right\}_{i=1}^{N_B}$ denote the document embeddings in domains A and B, respectively, we thus have

$$\boldsymbol{z}_i^{(A)} = \frac{\sum_{j=1}^{M+M_A} x_{ij}^{(A)} \tilde{\boldsymbol{z}}_j^{(A)}}{\sum_{j=1}^{M+M_A} x_{ij}^{(A)}}, i = 1, 2, \ldots, N_A, \quad (3)$$

$$\boldsymbol{z}_i^{(B)} = \frac{\sum_{j=1}^{M+M_B} x_{ij}^{(B)} \tilde{\boldsymbol{z}}_j^{(B)}}{\sum_{j=1}^{M+M_B} x_{ij}^{(B)}}, i = 1, 2, \ldots, N_B. \quad (4)$$

Letting $\mathbf{Z}_A$ and $\mathbf{Z}_B$ denote the $N_A \times k$ and $N_B \times k$ document embedding matrices, the above formulations can be converted to their matrix presentation, given as

$$\mathbf{Z}_A = \mathbf{D}^{-1}(\mathbf{X}_A)\mathbf{X}_A\widetilde{\mathbf{Z}}_A, \quad (5)$$

$$\mathbf{Z}_B = \mathbf{D}^{-1}(\mathbf{X}_B)\mathbf{X}_B\widetilde{\mathbf{Z}}_B, \quad (6)$$

where the notation $\mathbf{D}^{-1}(\mathbf{X})$ denotes a diagonal matrix computed from an input matrix $\mathbf{X} = [x_{ij}]$ with its i-th diagonal element obtained by $\left(\sum_j x_{ij}\right)^{-1}$.

Although a document is represented using the words that appear in that document, we are free to select any feature representation for the individual words. Specifically, the d features we use in the domain A (or the h features we use in the domain B) need not be words limited to the domain and can be, for example, bigrams of words or part-of-speech tags of the words. The document feature space and the word feature spaces are connected via Eqs. (3) and (4). This de-coupling of document and word representations allows us to incorporate semantically rich word representation such as the recently developed neural word embeddings [27], [28]. For the simplicity of the presentation, we limit the discussion in this paper to lexical features (unigrams and bigrams of words) and differ a study of rich semantic feature spaces to future work.

4.2 Model Construction

According to Eqs. (1), (2), (5) and (6), the computation of the word and document embeddings relies on the computation of the two projection matrices of $\mathbf{P}_A$ and $\mathbf{P}_B$ based on the input matrices of $\mathbf{U}_A$, $\mathbf{U}_B$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}_A$, $\mathbf{X}_B$ and $\mathbf{Y}$. In the following, we show how to derive $\mathbf{P}_A$ and $\mathbf{P}_B$ by solving an optimisation problem constructed based on the three rules.

4.2.1 Rule 1 based Modelling

Rule 1 aims at mapping the two embedded points $\boldsymbol{u}_i^{(A)}$ and $\boldsymbol{u}_i^{(B)}$ that represent the same pivot but are mapped from different domains as close as possible in $\mathbb{R}^k$. This is equivalent to minimising the difference between their corresponding embedding vectors $\left\|\widetilde{\boldsymbol{z}}_i^{(A)} - \widetilde{\boldsymbol{z}}_i^{(B)}\right\|_2^2$, where $\|\cdot\|_2$ denotes the l_2-norm and it is equivalent to the Euclidean norm if the input is a vector. To minimise such differences for all the pivots, the following objective function can be constructed

$$\sum_{i=1}^{M} \left\|\widetilde{\boldsymbol{z}}_i^{(A)} - \widetilde{\boldsymbol{z}}_i^{(B)}\right\|_2^2. \quad (7)$$

Incorporating the linear projection mapping as in Eqs. (1) and (2) for pivots, Eq. (7) can be expressed in matrix representation of

$$\text{tr}\left(\begin{bmatrix} \mathbf{U}_A\mathbf{P}_A \\ \mathbf{U}_B\mathbf{P}_B \end{bmatrix}^T \mathbf{L}\left(\begin{bmatrix} \mathbf{0}_{M\times M} & \mathbf{I}_{M\times M} \\ \mathbf{I}_{M\times M} & \mathbf{0}_{M\times M} \end{bmatrix}\right)\begin{bmatrix} \mathbf{U}_A\mathbf{P}_A \\ \mathbf{U}_B\mathbf{P}_B \end{bmatrix}\right). \quad (8)$$

The notation $\mathbf{I}_{M\times M}$ denotes an identity matrix of size M, $\mathbf{0}_{M\times N}$ denotes an $M \times N$ matrix with all elements equal to 0, and $\mathbf{1}_{M\times N}$ denotes an $M \times N$ matrix with all elements equal to one. In the above equation, $\mathbf{L}(\mathbf{W})$ defines the Laplacian matrix of an input square matrix of size M, given as $\mathbf{L}(\mathbf{W}) = \mathbf{D}(\mathbf{W}) - \mathbf{W}$, where $\mathbf{D}(\mathbf{W})$ is a diagonal matrix formed by the vector $\mathbf{W}\mathbf{1}_{M\times 1}$.

Defining a $(d+h) \times k$ matrix $\mathbf{P}$ so that $\mathbf{P}^T = [\mathbf{P}_A^T, \mathbf{P}_B^T]$, and a $2M \times (d+h)$ matrix $\mathbf{U}_1$ so that

$$\mathbf{U}_1 = \begin{bmatrix} \mathbf{U}_A & \mathbf{0}_{M\times h} \\ \mathbf{0}_{M\times d} & \mathbf{U}_B \end{bmatrix}, \quad (9)$$

we have

$$\begin{bmatrix} \mathbf{U}_A\mathbf{P}_A \\ \mathbf{U}_B\mathbf{P}_B \end{bmatrix} = \mathbf{U}_1\mathbf{P}. \quad (10)$$

Eq. (8) can then be formulated as a function of $\mathbf{P}$ as

$$O_1(\mathbf{P}) = \text{tr}\left(\mathbf{P}^T\mathbf{U}_1^T\mathbf{L}(\mathbf{W}_1)\mathbf{U}_1\mathbf{P}\right), \quad (11)$$

where the weight matrix is defined as

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{0}_{M\times M} & \mathbf{I}_{M\times M} \\ \mathbf{I}_{M\times M} & \mathbf{0}_{M\times M} \end{bmatrix}. \quad (12)$$

4.2.2 Rule 2 based Modelling

Rule 2 aims at minimising the friend closeness (C_F) while maximising the enemy dispersion (D_E) in the embedded space $\mathbb{R}^k$ for the labeled documents in domain A. Here, we refer documents that are within a certain neighbourhood of each other and belong to the same class as friends, while documents that are neighbours of each other but belong to different classes as enemies.

To identify the relationships between the N_{lA} labeled documents in terms of friend or enemy, we first construct an $N_{lA} \times N_{lA}$ similarity matrix $\mathbf{S}_{lA} = \left[s_{ij}^{(lA)}\right]$ based on their original feature representation stored in the $N_{lA} \times$

$(M+M_A)$ matrix $\mathbf{X}_{lA}=\left[x_{ij}^{(lA)}\right]$, for example, using the cosine similarity measure:

$$s_{ij}^{(lA)}=\frac{\sum_{k=1}^{M+M_A}x_{ik}^{(lA)}x_{jk}^{(lA)}}{\sqrt{\sum_{k=1}^{M+M_A}\left|x_{ik}^{(lA)}\right|^2}\sqrt{\sum_{k=1}^{M+M_A}\left|x_{jk}^{(lA)}\right|^2}} \tag{13}$$

Then, an $N_{lA}\times N_{lA}$ graph is constructed by applying K-nearest-neighbour (K-NN) search to $\mathbf{S}_{lA}$. Its corresponding adjacencies $\{\delta_{ij}\}_{i,j=1}^{N_{lA}}$ are functions of the neighbour number K, similarity matrix $\mathbf{S}_{lA}$ and the class matrix $\mathbf{Y}$, defined as follows

$$\delta_{ij}(K,\mathbf{S}_{lA},\mathbf{Y})=\begin{cases}+1, & \text{if } \boldsymbol{x}_i^{(lA)} \text{ and } \boldsymbol{x}_j^{(lA)} \text{ are from the same class and are undirected } K\text{-NNs},\\ -1, & \text{if } \boldsymbol{x}_i^{(lA)} \text{ and } \boldsymbol{x}_j^{(lA)} \text{ are from different classes and are undirected } K\text{-NNs},\\ 0, & \text{otherwise.}\end{cases} \tag{14}$$

The adjacency $\delta_{ij}=1$ indicates the ith and jth documents are friends of each other, $\delta_{ij}=-1$ for enemy, while $\delta_{ij}=0$ implies neither friend nor enemy relationship exists between the two documents.

Next, we define the following measures of friend closeness C_F and and enemy dispersion D_E to evaluate the overall distances between the friend and enemy documents in the embedded space $\mathbb{R}^k$, respectively,

$$C_F=\sum_{\delta_{ij}(K,\mathbf{S}_{lA},\mathbf{Y})=1}s_{ij}^{(lA)}\left\|\boldsymbol{z}_i^{(lA)}-\boldsymbol{z}_j^{(lA)}\right\|^2, \tag{15}$$

$$D_E=\sum_{\delta_{ij}(K,\mathbf{S}_{lA},\mathbf{Y})=-1}s_{ij}^{(lA)}\left\|\boldsymbol{z}_i^{(lA)}-\boldsymbol{z}_j^{(lA)}\right\|^2, \tag{16}$$

where $\left\{\boldsymbol{z}_i^{(lA)}\right\}_{i=1}^{N_{lA}}$ denotes the set of embedding vectors of the labeled documents in domain A. The above two quantities assess the overall distances weighted by the similarity values between friends and between enemies. According to rule 2, C_F is to be minimised while D_E to be maximised. This enables more similar friends with larger weights to be forced to come even closer in the embedded space to improve within-class compactness, while more similar enemies with larger weights that are very likely to be boundary points to be forced to stay further away from each other in the embedded space to improve the overall between-class separation. We maximise $D_E-\lambda_1 C_F$ to achieve the minimisation of C_F and the maximisation of D_E, where the weight parameter $\lambda_1>0$ controls the balance between D_E and C_F. After incorporating Eqs. (15) and (16), $D_E-\lambda_1 C_F$ can be written as

$$\text{tr}\left(\mathbf{Z}_{lA}^T\mathbf{L}(\mathbf{W}_2)\mathbf{Z}_{lA}\right), \tag{17}$$

where $\mathbf{Z}_{lA}$ denotes the embedding matrix of the labeled documents in domain A. Each element of the $N_{lA}\times N_{lA}$ weight matrix $\mathbf{W}_2=\left[w_{ij}^{(2)}\right]$ is defined as

$$w_{ij}^{(2)}=\begin{cases}-\lambda_1 s_{ij}^{(lA)}, & \text{if } \boldsymbol{x}_i^{(lA)} \text{ and } \boldsymbol{x}_j^{(lA)} \text{ are from the same class and are undirected } K\text{-NNs},\\ s_{ij}^{(lA)}, & \text{if } \boldsymbol{x}_i^{(lA)} \text{ and } \boldsymbol{x}_j^{(lA)} \text{ are from different classes and are undirected } K\text{-NNs},\\ 0, & \text{otherwise.}\end{cases} \tag{18}$$

After incorporating Eq. (5) to compute the embeddings for the labeled documents and Eq. (1) to compute the projection-based word embeddings, Eq. (17) becomes

$$\text{tr}\left(\mathbf{P}_A^T\mathbf{F}_2\mathbf{P}_A\right), \tag{19}$$

where the $d\times d$ matrix $\mathbf{F}_2$ is given as

$$\mathbf{F}_2=[\mathbf{U}_A^T,\mathbf{A}^T]\mathbf{X}_{lA}^T\mathbf{D}^{-1}(\mathbf{X}_{lA})\mathbf{L}(\mathbf{W}_2)\mathbf{D}^{-1}(\mathbf{X}_{lA})\mathbf{X}_{lA}\begin{bmatrix}\mathbf{U}_A\\ \mathbf{A}\end{bmatrix}. \tag{20}$$

Introducing $\mathbf{P}^T=[\mathbf{P}_A^T,\mathbf{P}_B^T]$, Eq. (19) becomes

$$O_2(\mathbf{P})=\text{tr}\left(\mathbf{P}^T\mathbf{Q}_2\mathbf{P}\right), \tag{21}$$

where $\mathbf{Q}_2$ is a square matrix of size $d+h$ defined as

$$\mathbf{Q}_2=\begin{bmatrix}\mathbf{F}_2 & \mathbf{0}_{d\times h}\\ \mathbf{0}_{h\times d} & \mathbf{0}_{h\times h}\end{bmatrix}. \tag{22}$$

4.2.3 Rule 3 based Modelling

Rule 3 aims at preserving the local neighbour geometry between the unlabeled documents in the source domain A and between the documents in the target domain B. This is a standard task in spectral embedding [29], [30] achieved by minimising a penalised distance error that pulls neighbour points closer in the embedded space. Two error functions are constructed for the two domains, respectively, of which one is

$$\text{error}_A=\sum_{i,j=1}^{N_{uA}}w_{ij}^{(3,uA)}\left\|\boldsymbol{z}_i^{(uA)}-\boldsymbol{z}_j^{(uA)}\right\|^2, \tag{23}$$

where $\left\{\boldsymbol{z}_i^{(uA)}\right\}_{i=1}^{N_{uA}}$ denotes the set of embedding vectors of the unlabeled documents in domain A, the other is

$$\text{error}_B=\sum_{i,j=1}^{N_B}w_{ij}^{(3,B)}\left\|\boldsymbol{z}_i^{(B)}-\boldsymbol{z}_j^{(B)}\right\|^2, \tag{24}$$

defined for all the documents in domain B. The penalising weights $w_{ij}^{(3,uA)}$ and $w_{ij}^{(3,B)}$ represent degrees of similarity between documents in the two domains.

Letting the $N_{uA}\times N_{uA}$ and $N_B\times N_B$ proximity weight matrices $\mathbf{W}_3^{(A)}=\left[w_{ij}^{(3,uA)}\right]$ and $\mathbf{W}_3^{(B)}=\left[w_{ij}^{(3,B)}\right]$ model the local neighbour geometry between documents, they can be constructed as follows: First, we construct an $N_{uA}\times N_{uA}$ similarity matrix $\mathbf{S}_{uA}=\left[s_{ij}^{(uA)}\right]$ between the unlabeled documents in domain A based on their original feature representation $\mathbf{X}_{uA}$ using the same cosine similarity measure as shown in Eq. (13). Similarly, an

$N_B \times N_B$ similarity matrix $\mathbf{S}_B = \left[s_{ij}^{(B)}\right]$ is constructed for all the documents in domain B by applying the cosine similarity measure over $\mathbf{X}_B$. Then, $\mathbf{W}_3^{(uA)}$ and $\mathbf{W}^{(B)}$ are constructed from $\mathbf{S}_{uA}$ and $\mathbf{S}_B$ respectively by applying the K-NN search such that

$$w_{ij}^{(3,uA)} = \begin{cases} s_{ij}^{(uA)}, & \text{if } \boldsymbol{x}_i^{(uA)} \text{ and } \boldsymbol{x}_j^{(uA)} \text{ are undirected } K\text{-NNs,} \\ 0, & \text{otherwise,} \end{cases} \tag{25}$$

and

$$w_{ij}^{(3,B)} = \begin{cases} s_{ij}^{(B)}, & \text{if } \boldsymbol{x}_i^{(B)} \text{ and } \boldsymbol{x}_j^{(B)} \text{ are undirected } K\text{-NNs,} \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Simultaneous minimisation of both errors in Eqs. (23) and (24) leads to the minimisation of the following joint error function of

$$\begin{aligned} &\text{error}_A + \lambda_2 \text{error}_B \\ &= \text{tr}\left(\mathbf{Z}_{uA}^T \mathbf{L}\left(\mathbf{W}_3^{(uA)}\right)\mathbf{Z}_{uA}\right) + \lambda_2 \text{tr}\left(\mathbf{Z}_B^T \mathbf{L}\left(\mathbf{W}_3^{(B)}\right)\mathbf{Z}_B\right), \end{aligned} \tag{27}$$

where the weight parameter $\lambda_2 > 0$ balances between the geometry measures in the two domains. After incorporating Eqs. (5) and (6) for computing document embeddings and Eqs. (1) and (2) for computing word embeddings, Eq. (27) can be written as

$$\text{tr}\left(\mathbf{P}_A^T \mathbf{F}_3^{(A)} \mathbf{P}_A\right) + \lambda_2 \text{tr}\left(\mathbf{P}_B^T \mathbf{F}_3^{(B)} \mathbf{P}_B\right). \tag{28}$$

Here, the $d \times d$ matrix $\mathbf{F}_3^{(A)}$ and the $h \times h$ matrix $\mathbf{F}_3^{(B)}$ are computed by

$$\mathbf{F}_3^{(A)} = \left[\mathbf{U}_A^T, \mathbf{A}^T\right] \mathbf{X}_{uA}^T \mathbf{D}^{-1}(\mathbf{X}_{uA}) \mathbf{L}\left(\mathbf{W}_3^{(A)}\right) \mathbf{D}^{-1}(\mathbf{X}_{uA}) \mathbf{X}_{uA} \begin{bmatrix} \mathbf{U}_A \\ \mathbf{A} \end{bmatrix},$$

$$\mathbf{F}_3^{(B)} = [\mathbf{U}_B^T, \mathbf{B}^T] \mathbf{X}_B^T \mathbf{D}^{-1}(\mathbf{X}_B) \mathbf{L}\left(\mathbf{W}_3^{(B)}\right) \mathbf{D}^{-1}(\mathbf{X}_B) \mathbf{X}_B \begin{bmatrix} \mathbf{U}_B \\ \mathbf{B} \end{bmatrix}.$$

Introducing $\mathbf{P}^T = \left[\mathbf{P}_A^T, \mathbf{P}_B^T\right]$, Eq. (28) becomes

$$O_3(\mathbf{P}) = \text{tr}\left(\mathbf{P}^T \mathbf{Q}_3 \mathbf{P}\right), \tag{29}$$

where $\mathbf{Q}_3$ is a composite square matrix of size $d + h$ defined as

$$\mathbf{Q}_3 = \begin{bmatrix} \mathbf{F}_3^{(A)} & \mathbf{0}_{d\times h} \\ \mathbf{0}_{h\times d} & \lambda_2 \mathbf{F}_3^{(B)} \end{bmatrix}. \tag{30}$$

4.2.4 Joint Optimisation

In order to simultaneously apply all the three rules in the embedded space, we construct a composite optimisation model that minimises O_1 and O_3, also maximises O_2, which leads to the following maximisation objective function:

$$O(\mathbf{P}) = w_2 O_2(\mathbf{P}) - w_1 O_1(\mathbf{P}) - w_3 O_3(\mathbf{P}) \tag{31}$$

$$= \text{tr}\left(\mathbf{P}^T \mathbf{Q} \mathbf{P}\right), \tag{32}$$

where the objective weights $0 \leq w_1, w_2, w_3 \leq$ modify the contributions of the three objective functions of O_1, O_2 and O_3, and $w_1 + w_2 + w_3 = 1$. The $(d + h) \times (d + h)$ objective matrix $\mathbf{Q}$ is computed by

$$\mathbf{Q} = \begin{bmatrix} w_2 \mathbf{F}_2 - w_3 \mathbf{F}_3^{(A)} & \mathbf{0}_{d\times h} \\ \mathbf{0}_{h\times d} & -w_3 \lambda_2 \mathbf{F}_3^{(B)} \end{bmatrix} - w_1 \mathbf{U}_1^T \mathbf{L}(\mathbf{W}_1) \mathbf{U}_1. \tag{33}$$

Usually, when multi-dimensional embeddings are computed, it is important to ensure independence between different embedding dimensions to maximise the information offered by the multi-dimensional space and to avoid redundancy between dimensions. In our case, each embedding dimension is induced by a projection vector corresponding to one column of the projection matrix $\mathbf{P}$. Thus, we enforce the orthogonality constraint $\mathbf{P}^T\mathbf{P} = \mathbf{I}_{k\times k}$ to achieve linear independence between different dimensions. Subsequently, this leads to the following constrained optimisation problem:

$$\max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}_{k\times k}} \text{tr}\left(\mathbf{P}^T \mathbf{Q} \mathbf{P}\right). \tag{34}$$

The optimal solution of this constrained optimisation problem corresponds to the top k eigenvectors of $\mathbf{Q}$ with the largest k eigenvalues [30].

The computational complexity of the proposed method is dominated by the eigenvalue decomposition of the matrix $\boldsymbol{Q}$. The computational complexity of the eigenvalue decomposition of an $n \times n$ symmetric square matrix $\boldsymbol{Q}$ is in general $\mathcal{O}(n^3)$ [31]. However, we do not require the full eigenvalue decomposition of $\boldsymbol{Q}$, but only the largest k eigenvectors. Truncated methods that require only $\mathcal{O}(kn^2)$ can be used in this case [32]. Moreover, for larger matrices, stochastic methods that approximately compute eigenvalue decomposition in $\mathcal{O}(n^2 \log(k) + 2nk^2)$ can be used [33]. We note that the computational complexity of the proposed method is comparable to SCL or SFA which are based on respectively SVD and eigenvalue decomposition.

4.3 Model Training

The above model involves the computation of three square similarity matrices of $\mathbf{S}_{lA}$, $\mathbf{S}_{uA}$ and $\mathbf{S}_B$ with the corresponding sizes of N_{lA}, N_{uA} and N_B. It also involves the K-NN search for each of the three similarity matrices. In the end, it requires the eigen-decomposition of the $(d+h)\times(d+h)$ matrix $\mathbf{Q}$, which is not necessarily sparse. Five parameters are to be set, including the neighbour number K, and four weight parameters $\lambda_1, \lambda_2 > 0$ and $0 < w_1, w_2 < 1$ (another weight parameter can be computed by $w_3 = 1 - w_1 - w_2$). If the three similarity matrices of $\mathbf{S}_{lA}$, $\mathbf{S}_{uA}$ and $\mathbf{S}_B$ are computed by measures involving extra parameters, e.g., polynomial kernel rather than cosine similarity, more parameters would need to be selected.

For a given source and target domain pair, we find the neighbourhood sizes K for the similarity matrices using held-out data that we set aside from the training dataset. The effect of K and the dimensionality of the

embedding on the performance of the proposed method is empirically studied in Section 6. We consider the unweighted combinations of the different objective functions. Specifically, we set one or more variables among w_1, w_2, and w_3 to zero, and measure the performance of domain adaptation. An extensive study of all possible combinations of weights is beyond the scope of this work. Because of its popularity and simplicity, we use cosine similarity as the similarity measure for finding nearest neighbours when creating the matrices $\mathbf{S}_{lA}$, $\mathbf{S}_{uA}$ and $\mathbf{S}_B$. We use NumPy[1], a numeric Python library, for eigenvalue decomposition. The source code and the data used in the paper will be publicly released. Logistic regression with l_2 regularisation is used as the binary sentiment classifier. We use the implementation of logistic regression in scikit-learn[2] in our experiments. l_2 regularisation coefficient is set to 1 throughout the experiments described in the paper.

5 Dataset

We use the cross-domain sentiment classification dataset[3] prepared by Blitzer et al. [10] in our experiments. This dataset consists of Amazon product reviews for four different product types: books, DVDs, electronics and kitchen appliances. Each review is assigned with a rating (0-5 stars), a reviewer name and location, a product name, a review title and date, and the review text. Reviews with rating > 3 are labeled as positive, whereas those with rating < 3 are labeled as negative. For each domain, there are 1000 positive and 1000 negative examples, the same balanced composition as the polarity dataset constructed by Pang et al. [34]. The dataset also contains on average $17,547$ unlabeled reviews for the four domains. Following previous work, we randomly select 800 positive and 800 negative labeled reviews from each domain as training instances (total number of training instances are $1600 \times 4 = 6400$), and the remainder is used for testing (total number of test instances are $400 \times 4 = 1600$).

Mutual information between a feature and the labeled reviews have been used in [10] for selecting pivots. However, in unsupervised domain adaptation we have labeled data only for the source domain. Therefore, there is no guarantee that we will obtain pivots that behave similarly in both the source as well as the target domains by this method. Moreover, source domain labeled data are only a fraction of all the data available for the adaptation task. Co-occurrence counts and probability estimates conducted using small datasets are likely to be sparse and unreliable. To overcome these issues, we use a pointwise mutual information (PMI)-based pivot selection method to select pivots that consider both source and the target domains. Specifically, we measure the PMI between a feature w in a domain and the label (name) of that domain as follows,

$$\mathrm{PMI}(w, \mathcal{D}_l) = \frac{p(w|\mathcal{D}_l)}{p(w)} = \log\left(\frac{h(w, \mathcal{D}_l)N}{h(w)h(\mathcal{D}_l)}\right). \quad (35)$$

Here, $\mathcal{D}_l$ denotes the label (name) of the domain. For example, for the *books* domain we consider *book* as the domain label. $h(w, \mathcal{D}_l)$ is the total number of documents in which w and $\mathcal{D}_l$ co-occurred, $h(w)$ is the total number of documents in which w occur, and $h(\mathcal{D}_l)$ is the total number of documents in which the domain label $\mathcal{D}_l$ occur. N is the total number of documents in the domain $\mathcal{D}$. For a given source domain $\mathcal{S}$ and target domain $\mathcal{T}$, we sort features w in the descending order of $\mathrm{PMI}(w, \mathcal{S}_l) + \mathrm{PMI}(w, \mathcal{T}_l)$, and select the top ranked features as pivots in the proposed method. Intuitively, for a particular feature to get ranked high enough to be selected as a pivot under this method, it must be closely associated to both the source and the target domains. Indeed, manual inspection of the pivots selected for various pairs of domains reveal that the above-mentioned procedure accurately selects pivots that are domain independent.

6 Experiments and Results

In our experiments, we select 500 pivots (ie. $M = 500$) for each pair of domains. We select most co-occurring 1000 features (unigrams and bigrams) from each domain to represent pivots (ie. $d = h = 1000$). Rules 2 and 3 have neighbourhoods (K) and combination coefficients (λ_1 and λ_2) as parameters. We later discuss the effect of parameters on each rule.

In Table 3, we show the classification accuracy on the target domain reviews for the individual rules (denoted by $R1$, $R2$, and $R3$), and their combinations (denoted by $+$ signs) for different source domains. For succinctness in Table 3, we denote the four domains *books*, *DVDs*, *electronics*, and *kitchen appliances* respectively by letters **B**, **D**, **E**, and **K**. We create 30 dimensional embeddings in all those combinations. **No Adapt** baseline shows the level of performance we would obtain if we use the classifier trained on the source domain labeled documents without any domain adaptation on the target domain. Therefore, **No Adapt** can be considered as a lower-baseline that denotes the level of performance we could expect if we did not perform any domain adaptations.

We use the same set of 500 pivots we selected for the proposed method as the domain independent features required by **SFA**. The dimensionality of the embeddings created by SFA is set to 30. SFA is considered as the current state-of-the-art cross-domain sentiment classification method. We use the same 500 pivots as used by the proposed method and **SFA** to train linear pivot predictor required by **SCL**. The dimensionality of the feature space created by the SVD step in SCL is set to 30.[4]

1. www.numpy.org
2. http://scikit-learn.org/
3. http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

4. We observed that the performance of SCL is largely unaffected over a range of dimensionalities within $[10, 100]$.

Although Amazon review dataset that we use in this work has been used frequently in prior evaluations, the differences in pre-processing (lemmatisation, unigram/bigram extraction, pivot selection, feature weighting, vector normalisation via l_1/l_2), differences in train/test splits (different random splits), and differences in classification algorithms used for sentiment classification (SVM, logistic regression etc. and their regularisation parameters) make direct comparisons of results reported across different publications difficult. Therefore, to conduct a fair comparison, we fix all those settings across different methods, and evaluate under the same conditions. We use the same feature space and pivot set for all methods, and use a logistic regression classifier with l_2 regularisation coefficient set to 1.0 as the sentiment classifier with all the methods compared in Table 3. Because the set of pivots, dimensionality of the embeddings created, and the binary sentiment classifier trained are all equal among the different methods compared in Table 3, the differences in performance reported in Table 3 can be directly attributable to the embeddings created by those methods.

For each pair of source and target domains in the benchmark dataset, we show the best classification accuracies obtained by a method on the target domain test reviews in boldface in Table 3. As an upper baseline, we train and test a sentiment classifier using target domain data. Although this is not a domain adaptation setting, it demonstrates the level of performance we can expect to achieve if we had labeled data for the target domain. The respective accuracies on the four domains are: *books* (0.8010), *electronics* (0.8388), *DVDs* (0.7954), and *kitchen* (0.8442). From Table 3 we see that out of the 12 pairs of domains, some combination of the proposed embeddings report the best results for 9 pairs of domains. Best results are obtained by **SFA**, for the three domain pairs: **E-B**, **K-B**, and **D-K**. However, the differences in performance by the proposed method and SFA is not statistically significant according to the binomial exact test under 0.05 confidence level.

Among the three rules we proposed, **R1** performs poorly on its own. Recall that **R1** is concerned with projecting pivots in the two domains into closer locations in the embedded space. This rule is agnostic to sentiment information available only in the labels assigned to the source reviews. Therefore, it is not surprising that using **R1** alone results in poor performance. Comparatively, **R2**, which attempts to enforce the label information available in the source domain reviews performs better than **R1** in all cases. This shows the importance of using label information available for the final task (i.e. sentiment classification), when performing the auxiliary tasks such as embedding learning. **R3** considers the target domain unlabeled reviews which is the only source of information regarding the target domain in which the trained sentiment classifier will be ultimately evaluated upon. Among **R1**, **R2**, and **R3**, we see that **R3** reports the best performance in all 12 domain pairs. This result demonstrates the differences between the source and the target domains, and the need for domain adaptation methods. Simply considering all the data available in the source domain (both labeled reviews and unlabeled reviews) alone is insufficient to obtain good performances in the target domain, and we must consider the target domain reviews when creating embeddings.

Among the three pairwise combinations **R1+R2**, **R1+R3**, and **R2+R3**, we see that by combining **R1** which performed poorly on its own with **R3**, we get better results than when **R1** is combined with **R2**. This trend can be observed for all 12 domain pairs. Among all the combinations we compare in Table 3, **R2+R3** reports best results for the most number of domain pairs (6 out of 12). Considering that **R2** captures labeled information for the source domain and **R3** captures the geometry in the target domain, the combination of the two rules performing best in most cases supports our proposal to use the labeled reviews for the source domain when learning embeddings for domain adaptation.

The combination of the three rules **R1+R2+R3** does not surpass the best result obtained using only a single rule. Note that the overall objective function given by Eq. 31 is the linearly weighted sum of the three objective functions corresponding to the three rules. There exist only a small amount of source domain labeled reviews for training (e.g. 400 for a pair of domains in the benchmark dataset), compared to the large number of unlabeled reviews. Our preliminary studies show that balancing the objective functions by the number of documents or considering different linear combinations of the objectives other than the $0/1$ combinations shown in Table 3 is insufficient to produce a combination that consistently outperform the best results obtained by any single rule. An extensive study of the optimal combination of different rules is beyond the scope of this work, and is differed to future work.

An interesting observation in Table 3 is that domain adaptation is not a symmetric process. For example, when we adapt from the *books* domain to the *kitchen* domain, we obtain an accuracy of 0.7736 for the **R1+R2+R3** approach, whereas the same for the reverse adaptation task is as low as 0.6649. Selecting the correct source domain to adapt to a given target domain is an important decision. For example, if we would like to train a sentiment classifier for the *kitchen* domain, it is best to use *electronics* as the source domain. Considering that there are many electronic appliances such as ovens, microwaves, blenders, etc. that are used in the kitchen, we can expect a high degree of similarity between the *electronics* and *kitchen* domains, making it easier to adapt from *electronics* domain to *kitchen* domain. Theoretical and empirical studies have shown that the similarity between two domains (e.g. measured using α divergence) is a factor related to the ease of adaptation [35]. However, in practice, we encounter the situation that we have thousands of source domains available and selecting the best source domain to adapt to a given target domain remains a

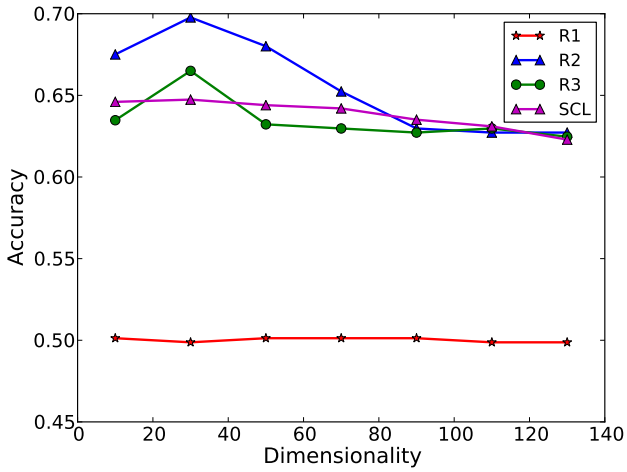

Fig. 1: Accuracy vs. the dimensionality of the embedding for *DVDs* source and *electronics* target domains.

challenging task. Prior work on cross-domain sentiment classification have been limited to pairwise adaptation (e.g. one source domain to one target domain), except for a notable few that use a collection of source domains [13], [14]. In certain circumstances, adapting from a particular source domain has shown to result in performance lower than not adapting. This phenomenon is referred to as *negative transfer*, and makes the task of finding suitable source domains for adapting to a given target domain a difficult one [36].

For rule 2, we study the effect of the two parameters neighbourhood K for the k-NN step, and the parameter λ_1 that balances the influence between friend closeness and enemy dispersion factors. Table 4 shows the classification accuracy for three target domains when the *books* domain is used as the source. We experimented with three values for the neighbourhood K (i.e.. 1, 5, 9), and three values for λ_1 (i.e. 0, 1, 10000). $\lambda_1 = 0$ simulates the scenario where we only consider enemy dispersion in rule 2. On the other hand, $\lambda_1 = 10000$ assigns a very high emphasis on friend closeness, thereby effectively neglecting the enemy dispersion. $\lambda_1 = 1$ gives an equal weight to enemy dispersion and friend closeness factors. From Table 4 we see that overall the classification accuracy on target domain test reviews decreases when larger K values are used when constructing the k-NN graph. Recall that in Rule 2, we are considering labeled documents. Because documents with low similarity scores get attached when the neighbourhood size is increased, it results in poor embeddings. This result shows that it is sufficient to consider smaller K values in practice for Rule 2. Smaller K values are desirable from a computational efficiency point of view because smaller K values often result in sparser k-NN graphs, which can be conveniently stored and processed even when the number of source domain labeled documents increases. On the other hand, the parameter λ_1 has a lesser influence compared to the neighbourhood K. In fact, when we consider smaller neighbourhoods such as when $K = 1$, we see that λ_1 does not affect the performance of the proposed method. Although for the limited availability of space we showed the results when the *books* domain is used as the source in Table 4, similar trends were observed for all domain pairs in the benchmark dataset.

Table 5 shows the effect of the neighbourhood size K and parameter λ_2 on the performance of Rule 3. Value ranges for K and λ_2 are set to the same ranges as for Rule 2 in Table 4. We see that increasing the size of the neighbourhood K reduces performance when $\lambda_2 = 0$ or 1. However, the trend is reversed for $\lambda_2 = 10000$. Recall that λ_2 is the relative weight assigned to the objective function corresponding to the target domain unlabeled reviews. Higher λ_2 values emphasise preserving the geometry in the target domain's feature space compared to that in the source domain. Therefore, when we increase the neighbourhood size in the $\lambda_2 = 10000$ setting we are considering similar documents from the target domain in which we will be applying the trained sentiment classifier eventually. Therefore, increasing the neighbourhood size in Rule 3 has a positive effect when target domain reviews are considered.

We study the relationship between the dimensionality of the embeddings we learn using the proposed method and the accuracy obtained on the target domain. The performance obtained using *DVDs* as the source domain and *electronics* as the target domain is shown in Figure 1. From Figure 1 we see that the performances of rules 2 and 3 reach a peak at 30 dimensions and then decreases and saturates. Embeddings with smaller dimensionalities are insufficient to capture the information in the source and target domains. On the other hand, we must have a sufficiently abstract representation of data in order to be able to transfer the knowledge we learn from one domain to another. We believe the trade-off of these two factors appears as a peak point in Figure 1. Although the exact value of the peak point shown in Figure 1 differs from one pair of domains to another, similar trends could be observed for all domain pairs. Rule 1 shows almost 0.5 classification accuracy irrespective of the dimensionality of the embeddings used. This result shows that we cannot learn a good sentiment classifier by using an embedding learnt purely from the pivots.

We show the nearest neighbours measured by the cosine similarity (shown within parenthesis) in the source and the target domains for the embeddings learnt from the three rules for some randomly selected words in Table 6. Bigrams are denoted by a + sign. We see that in the original space the nearest neighbours are not necessarily semantically related to the target word. However, in the embeddings learnt using rules 1, 2, and 3 contain some related features such as *fact* of a *story*, *bad points*, *read* and *remember*. Note that rule 2 considers features only for the source domain, thus the target domain projection matrix learnt using rule 2 is always zero (i.e. $\mathbf{P}_b = \mathbf{0}$). Therefore,

TABLE 3: Classification accuracies on the target domain when different source domains are used.

source	target	R1	R2	R3	R1+R2	R1+R3	R2+R3	R1+R2+R3	No Adapt	SFA	SCL
B	D	0.5063	0.6879	**0.7212**	0.5063	0.6828	**0.7212**	0.6828	0.6086	0.6445	0.6803
B	E	0.5012	0.6826	**0.7027**	0.5012	**0.7027**	**0.7027**	**0.7027**	0.6221	0.6625	0.6675
B	K	0.5	0.7060	0.7311	0.5	**0.7336**	0.7311	**0.7336**	0.6532	0.7161	0.6834
E	B	0.5012	0.6221	0.6246	0.5012	0.6246	0.6246	0.6246	0.5919	**0.6474**	0.6045
E	K	0.5	0.7914	**0.8065**	0.5	0.8040	**0.8065**	0.8040	0.6658	0.6256	0.7462
E	D	0.5063	0.6675	**0.6828**	0.5063	**0.6828**	**0.6828**	**0.6828**	0.5831	0.6266	0.6164
K	B	0.5037	0.6448	0.6649	0.5037	0.6649	0.6649	0.6649	0.6246	**0.6675**	0.6599
K	E	0.5012	**0.7405**	**0.7405**	0.5012	0.7380	**0.7405**	0.7380	0.6675	0.6096	0.7053
K	D	0.5063	0.6803	**0.7007**	0.5063	**0.7007**	**0.7007**	**0.7007**	0.6138	0.6419	0.6496
D	B	0.5012	**0.7052**	0.7002	0.7103	0.7002	0.7002	0.7002	0.5994	0.5919	0.6398
D	E	0.5012	0.6952	0.7002	0.5012	**0.7052**	0.7002	**0.7052**	0.6347	0.6877	0.6474
D	K	0.5	0.7236	0.7311	0.5	0.7311	0.7311	0.7311	0.6859	**0.7362**	0.6683
Average		0.5024	0.6956	0.7089	0.5198	0.7059	0.7089	0.7059	0.6291	0.6547	0.6604

TABLE 4: Effect of the neigbourhood size K and the parameter λ_1 on the classification accuracy of rule 2.

K	λ_1	B-D	B-E	B-K
1	0	0.6879	0.6725	0.6758
5	0	0.6445	0.6498	0.6783
9	0	0.6393	0.5919	0.5979
1	1	0.6879	0.6725	0.6758
5	1	0.6572	0.6397	0.6356
9	1	0.6496	0.6523	0.6231
1	10000	0.6879	0.6725	0.6758
5	10000	0.6521	0.6372	0.6306
9	10000	0.6521	0.6221	0.6206

TABLE 5: Effect of the neighbourhood size K and the parameter λ_2 on the classification accuracy of Rule 3.

K	λ_2	B-D	B-E	B-K
1	0	0.6828	0.6675	0.6758
5	0	0.6393	0.6246	0.6080
9	0	0.6675	0.6070	0.63316
1	1	0.6828	0.6675	0.6758
5	1	0.6803	0.6523	0.6783
9	1	0.6803	0.6397	0.6809
1	10000	0.6828	0.6675	0.6758
5	10000	0.7109	0.6826	0.7211
9	10000	0.7109	0.6826	0.7211

no neighbours in the target domain are listed for rule 2 in Table 6.

7 Conclusion

We considered three constraints that must be satisfied by an embedding that can be used to train a cross-domain sentiment classification method. We evaluated the performance of the individual constraints as well as their combinations using a benchmark dataset for cross-domain sentiment classification. Our experimental results show that some of the combinations of the proposed constraints obtain results that are statistically comparable to the current state-of-the-art methods for cross-domain sentiment classification. Unlike previously proposed embedding learning approaches for cross-domain sentiment classification, our proposed method uses the label information available for the source domain reviews, thereby learning embeddings that are sensitive to the final task of application, which is sentiment classification.

References

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[2] Y. Lu, C. Zhai, and N. Sundaresan, "Rated aspect summarization of short comments," in *WWW 2009*, 2009, pp. 131–140.

[3] T.-K. Fan and C.-H. Chang, "Sentiment-oriented contextual advertising," *Knowledge and Information Systems*, vol. 23, no. 3, pp. 321–344, 2010.

[4] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *KDD 2004*, 2004, pp. 168–177.

[5] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 2002.

[6] H. Daumé III, A. Kumar, and A. Saha, "Co-regularization based semi-supervised domain adaptation," in *NIPS'10*, 2010.

[7] D. Lopez-Paz, J. M. Hernandez-Lobato, and B. Scholkopf, "Semi-supervised domain adaptation with non-parametric copulas," in *NIPS'12*, 2012.

[8] H. Daumé III, "Frustratingly easy domain adaptation," in *ACL 2007*, 2007, pp. 256–263.

[9] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *EMNLP*, 2006, pp. 120 – 128.

[10] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *ACL 2007*, 2007, pp. 440–447.

[11] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *WWW 2010*, 2010, pp. 751 – 760.

[12] D. Bollegala, D. Weir, and J. Carroll, "Cross-domain sentiment classification using a sentiment sensitive thesaurus," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1719 – 1731, August 2013.

[13] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *ICML'11*, 2011.

TABLE 6: Nearest neighbours for words in source and target domains in the embedded space.

word	original		Rule 1		Rule 2		Rule 3	
	source	target	source	target	source	target	source	target
story	i+do (0.9817)	i+do (0.9866)	i+do (0.9999)	fact (0.9982)	fact (0.9973)	N/A	tell (0.9931)	fact (0.9983)
watch	set (0.9871)	say (0.9874)	set (0.9999)	get (0.9962)	set (0.9973)	N/A	expect (0.9921)	get (0.9968)
read	how (0.9820)	need (0.9893)	never (0.9999)	remember (0.9974)	some (0.9963)	N/A	perfect (0.9981)	mean (0.9973)
book	i+do (0.9834)	as+well (0.9878)	the+movie (0.9999)	be+really (0.9984)	fact (0.9962)	N/A	write (0.9721)	think (0.9904)
bad	when+i (0.9736)	be+great (0.9874)	point (0.9999)	point (0.9992)	not+have (0.9936)	N/A	can+not (0.9812)	point (0.9994)

[14] D. Bollegala, D. Weir, and J. Carroll, "Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification," in *ACL/HLT'11*, 2011, pp. 132 – 141.

[15] ——, "Learning to predict distributions of words across domains," in *Proc. of Association for Computational Linguistics (ACL)*, 2014, pp. 613 – 623.

[16] P. H. Calais Guerra, A. Veloso, W. Meira, Jr., and V. Almeida, "From bias to opinion: A transfer-learning approach to real-time sentiment analysis," in *Proc. of KDD*, 2011, pp. 150–158.

[17] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, no. 6, pp. 391–407, 1990.

[18] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 1986.

[19] X. He and P. Niyogi, "Locality preserving projections," in *NIPS*, 2003.

[20] E. Kokiopoulou and Y. Saad, "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2143 – 2156, December 2007.

[21] D. Cai and X. He, "Orthogonal locality preserving indexing," in *Proc. of 28th Annual International ACM Conference on Research and Development in Information Retrieval*, 2005.

[22] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179 – 188, 1936.

[23] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1027 – 1061, 2007.

[24] T. Mu, J. Y. Goulermas, J. Tsujii, and S. Ananiadou, "Proximity-based frameworks for generating embeddings from multi-output data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2216 – 2232, November 2012.

[25] M. Sugiyama, T. Ide, S. Nakajima, and J. Sese, "Semi-supervised local fisher discriminant analysis for dimensionality reduction," *Machine Learning*, vol. 78, no. 1/2, pp. 35 – 61, 2010.

[26] T. Mu, J. Jiang, Y. Wang, and J. Y. Goulermas, "Adaptive data embedding framework for multi-class classification," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1291–1303, 2012.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111 – 3119.

[28] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continous space word representations," in *NAACL'13*, 2013, pp. 746 – 751.

[29] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[30] T. Mu, J. Y. Goulermas, J. Tsujii, and S. Ananiadou, "Proximity-based frameworks for generating embeddings from multi-output data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2216–2232, 2012.

[31] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, MD: John Hopkins University Press, 1996.

[32] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *Journal of Machine Learning Research*, vol. 14, pp. 899 – 925, 2013.

[33] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: stochastic algorithms for constructing approximate matrix decompositions," California Institute of Technology, Tech. Rep., 2009.

[34] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *EMNLP 2002*, 2002, pp. 79–86.

[35] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, pp. 151–175, 2009.

[36] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345 – 1359, October 2010.

Danushka Bollegala received his PhD from the University of Tokyo. He is a senior lecturer in the Department of Computer Science at the University of Liverpool. His research interests include natural language processing, Web data mining, and machine learning. He is a member of Association for Computational Linguistics (ACL).

Tingting Mu received the BEng degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004, and the PhD degree in electrical engineering and electronics from the University of Liverpool, United Kingdom, in 2008. She is currently a lecturer in the School of Electrical Engineering, Electronics and Computer Science at the University of Liverpool. She is a member of the IEEE.

John Yannis Goulermas received the BSc degree (first class) in computation from the University of Manchester Institute of Science and Technology (UMIST), United Kingdom, in 1994, and the MSc and PhD degrees from the Control Systems Center, UMIST, in 1996 and 2000, respectively. He is currently a reader in the School of Electrical Engineering, Electronics and Computer Science at the University of Liverpool, United Kingdom. He is a senior member of the IEEE.