Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification

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

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

Abstract

In this paper, we study cross-domain sentiment classification with neural network architectures. We borrow the idea from Structural Correspondence Learning and use two auxiliary tasks to help induce a sentence embedding that supposedly works well across domains for sentiment classification. We also propose to jointly learn this sentence embedding together with the sentiment classifier itself. Experiment results demonstrate that our proposed joint model outperforms several state-of-the-art methods on five benchmark datasets.

1 Introduction

With the growing need of correctly identifying the sentiments expressed in subjective texts such as product reviews, sentiment classification has received continuous attention in the NLP community for over a decade (Pang et al., 2002; Pang and Lee, 2004; Hu and Liu, 2004; Choi and Cardie, 2008; Nakagawa et al., 2010). One of the big challenges of sentiment classification is how to adapt a sentiment classifier trained on one domain to a different new domain. This is because sentiments are often expressed with domain-specific words and expressions. For example, in the **Movie** domain, words such as *moving* and *engaging* are usually positive, but they may not be relevant in the **Restaurant** domain. Since labeled data is expensive to obtain, it would be very useful if we could adapt a model trained on a *source domain* to a *target domain*.

Much work has been done in sentiment analysis to address this *domain adaptation* problem (Blitzer et al., 2007; Pan et al., 2010; Bollegala et al., 2011; Ponomareva and Thelwall, 2012; Bollegala et al., 2016). Among them, an appealing method is the Structural Correspondence Learning (SCL) method (Blitzer et al., 2007), which uses pivot feature prediction tasks to induce a projected feature space that works well for both the source and the target domains. The intuition behind is that these pivot prediction tasks are highly correlated with the original task. For sentiment classification, Blitzer et al. (2007) first chose pivot words which have high mutual information with the sentiment labels, and then set up the pivot prediction tasks to be the predictions of each of these pivot words using the other words.

However, the original SCL method is based on traditional discrete feature representations and linear classifiers. In recent years, with the advances of deep learning in NLP, multi-layer neural network models such as RNNs and CNNs have been widely used in sentiment classification and achieved good performance (Socher et al., 2013; Dong et al., 2014a; Dong et al., 2014b; Kim, 2014; Tang et al., 2015). In these models, dense, real-valued feature vectors and non-linear classification functions are used. By using real-valued word embeddings pre-trained from a large corpus, these models can take advantage of the embedding space that presumably better captures the syntactic and semantic similarities between words. And by using non-linear functions through multi-layer neural networks, these models represent a more expressive hypothesis space. Therefore, it would be interesting to explore how these neural network models could be extended for cross-domain sentiment classification.

There has been some recent studies on neural network-based domain adaptation (Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2014). They use Stacked Denoising Auto-encoders (SDA) to induce a hidden representation that presumably works well across domains. However, SDA is fully unsupervised and does not consider the end task we need to solve, i.e., the sentiment classification task. In contrast, the idea behind SCL is to use carefully-chosen auxiliary tasks that correlate with the end task to induce a hidden representation. Another line of work aims to learn a low dimensional representation for each feature in both domains based on predicting its neighboring features (Yang and Eisenstein, 2015; Bollegala et al., 2015). Different from these methods, we aim to directly learn sentence embeddings that work well across domains.

In this paper, we aim to extend the main idea behind SCL to neural network-based solutions to sentiment classification to address the domain adaptation problem. Specifically, we borrow the idea of using pivot prediction tasks from SCL. But instead of learning thousands of pivot predictors and performing singular value decomposition on the learned weights, which all relies on *linear* transformations, we introduce only two auxiliary binary prediction tasks and directly learn a *non-linear* transformation that maps an input to a dense embedding vector. Moreover, different from SCL and the auto-encoder-based methods, in which the hidden feature representation and the final classifier are learned sequentially, we propose to jointly learn the hidden feature representation together with the sentiment classification model itself, and we show that joint learning works better than sequential learning.

We conduct experiments on a number of different source and target domains for sentence-level sentiment classification. We show that our proposed method is able to achieve the best performance compared with a number of baselines for most of these domain pairs.

2 Related Work

Domain Adaptation: Domain adaptation is a general problem in NLP and has been well studied in recent years (Blitzer et al., 2006; Daumé III, 2007; Jiang and Zhai, 2007; Dredze and Crammer, 2008; Titov, 2011; Yu and Jiang, 2015). For sentiment classification, most existing domain adaptation methods are based on traditional discrete feature representations and linear classifiers. One line of work focuses on inducing a general low-dimensional cross-domain representation based on the co-occurrences of domain-specific and domain-independent features (Blitzer et al., 2007; Pan et al., 2010; Pan et al., 2011). Another line of work tries to derive domain-specific sentiment words (Bollegala et al., 2011; Li et al., 2012). Our proposed method is similar to the first line of work in that we also aim to learn a general, cross-domain representation (sentence embeddings in our case).

Neural Networks for Sentiment Classification: A recent trend of deep learning enhances various kinds of neural network models for sentiment classification, including Convolutional Neural Networks (CNNs), Recursive Neural Network (ReNNs) and Recurrent Neural Network (RNNs), which have been shown to achieve competitive results across different benchmarks (Socher et al., 2013; Dong et al., 2014a; Dong et al., 2014b; Kim, 2014; Tang et al., 2015). Inspired by their success in standard in-domain settings, it is intuitive for us to apply these neural network models to domain adaptation settings.

Denoising Auto-encoders for Domain Adaptation: Denoising Auto-encoders have been extensively studied in cross-domain sentiment classification, since the representations learned through multi-layer neural networks are robust against noise during domain adaptation. The initial application of this idea is to directly employ stacked denoising auto-encoders (SDA) by reconstructing the original features from data that are corrupted with noise (Glorot et al., 2011), and Chen et al. (2012) proposed to analytically marginalize out the corruption during SDA training. Later Yang and Eisenstein (2014) further showed that their proposed structured dropout noise strategy can dramatically improve the efficiency without sacrificing the accuracy. However, these methods are still based on traditional discrete representation and do not exploit the idea of using auxiliary tasks that are related to the end task. In contrast, the sentence embeddings learned from our method are derived from real-valued feature vectors and rely on related auxiliary tasks.

3 Method

In this section we present our sentence embedding-based domain adaptation method for sentiment classification. We first introduce the necessary notation and an overview of our method. we then delve into the details of the method.

3.1 Notation and Method Overview

We assume that each input is a piece of text consisting of a sequence of words. For the rest of this paper, we assume each input is a sentence, although our method is general enough for longer pieces of text. Let $\boldsymbol{x} = (x_1, x_2, \ldots)$ denote a sentence where each $x_i \in \{1, 2, \ldots, V\}$ is a word in the vocabulary and V is the vocabulary size. Let the sentiment label of $\boldsymbol{x}$ be $y \in \{+, -\}$ where $+$ denotes a positive sentiment and $-$ a negative sentiment. We further assume that we are given a set of labeled training sentences from a source domain, denoted by $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N^s}$. Also, we have a set of unlabeled sentences from a target domain, denoted by $\mathcal{D}^t = \{\boldsymbol{x}_i^t\}_{i=1}^{N^t}$. Our goal is to learn a good sentiment classifier from both $\mathcal{D}^s$ and $\mathcal{D}^t$ such that the classifier works well on the target domain.

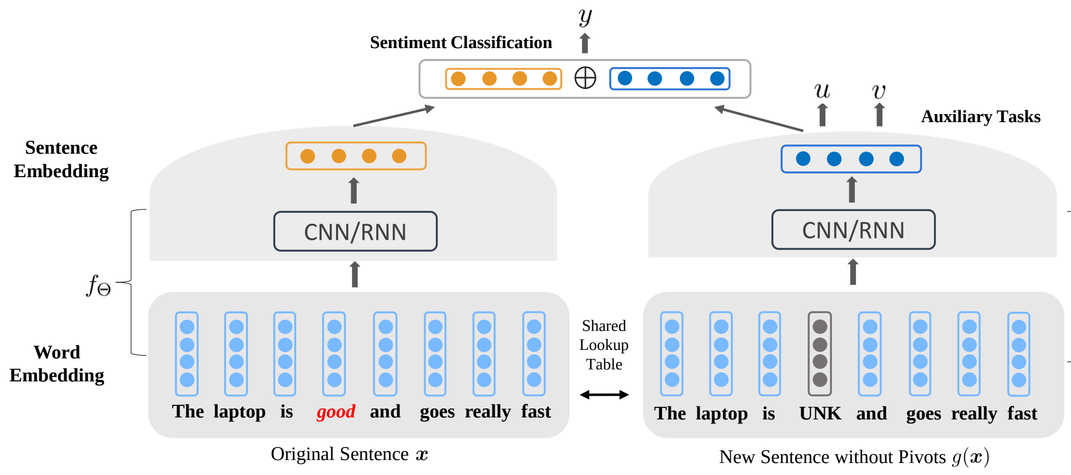
A baseline solution without considering any domain difference is to simply train a classifier using $\mathcal{D}^s$, and with the recent advances in neural network-based methods to sentence classification, we consider a baseline that uses a multi-layer neural network such as a CNN or an RNN to perform the classification task. To simplify the discussion and focus on the domain adaptation ideas we propose, we will leave the details of the neural network model we use in Section 3.5. For now, we assume that a multi-layer neural network is used to transform each input $\boldsymbol{x}$ into a sentence embedding vector $\boldsymbol{z}$. Let us use f_Θ to denote the transformation function parameterized by Θ, that is, $\boldsymbol{z} = f_\Theta(\boldsymbol{x})$. Next, we assume that a linear classifier such as a softmax classifier is learned to map $\boldsymbol{z}$ to a sentiment label y.

We introduce two auxiliary tasks which presumably are highly correlated with the sentiment classification task itself. Labels for these auxiliary tasks can be automatically derived from unlabeled data in both the source and the target domains. With the help of the two auxiliary tasks, we learn a non-linear transformation function $f_{\Theta'}$ from unlabeled data and use it to derive a sentence embedding vector $\boldsymbol{z}'$ from sentence $\boldsymbol{x}$, which supposedly works better across domains. Finally we use the source domain's training data to learn a linear classifier on the representation $\boldsymbol{z} \oplus \boldsymbol{z}'$, where $\oplus$ is the operator that concatenates two vectors. Figure 1 gives the outline of our method.

3.2 Auxiliary Tasks

Our two auxiliary tasks are about whether an input sentence contains a positive or negative domain-independent sentiment word. The intuition is the following. If we have a list of domain-independent positive sentiment words, then an input sentence that contains one of these words, regardless of the domain the sentence is from, is more likely to contain an overall positive sentiment. For example, a sentence containing the word *good* is likely to be overall positive. Moreover, the rest of the sentence excluding the word *good* may contain domain-specific words or expressions that also convey a positive sentiment. For example, in the sentence "The laptop is good and goes really fast," we can see that the word *fast* is a domain-specific sentiment word, and its sentiment polarity correlates with that of the word *good*, which is domain-independent. Therefore, we can hide the domain-independent positive words in a sentence and try to use the other words in the sentence to predict whether the original sentence contains a domain-independent positive word. There are two things to note about this auxiliary task: (1) The label of the task can be automatically derived provided that we have the domain-independent positive word list. (2) The task is closely related to the original task of sentence-level sentiment classification. Similarly, we can introduce a task to predict the existence of a domain-independent negative sentiment word in a sentence.

Formally, let us assume that we have two domain-independent sentiment word lists, one for the positive sentiment and the other for the negative sentiment. Details of how these lists are obtained will be given in Section 3.5. Borrowing the term from SCL, we refer to these sentiment words as pivot words. For each sentence $\boldsymbol{x}$, we replace all the occurrences of these pivot words with a special token *UNK*. Let $g(\cdot)$ be a function that denotes this procedure, that is, $g(\boldsymbol{x})$ is the resulting sentence with



y
Sentiment Classification
$\oplus$
u
v
Auxiliary Tasks
Sentence Embedding
CNN/RNN
CNN/RNN
f_Θ
$f_{\Theta'}$
Word Embedding
Shared Lookup Table
The laptop is *good* and goes really fast
The laptop is UNK and goes really fast
Original Sentence $\boldsymbol{x}$
New Sentence without Pivots $g(\boldsymbol{x})$

Figure 1: The Outline of our Proposed Method.

UNK tokens. We then introduce two binary labels for $g(\boldsymbol{x})$. The first label u indicates whether the original sentence $\boldsymbol{x}$ contains at least one domain-independent positive sentiment word, and the second label v indicates whether $\boldsymbol{x}$ contains at least one domain-independent negative sentiment word. Figure 1 shows an example sentence $\boldsymbol{x}$, its modified version $g(\boldsymbol{x})$ and the labels u and v for $\boldsymbol{x}$. We further use $\mathcal{D}^a = \{(\boldsymbol{x}_i, u_i, v_i)\}_{i=1}^{N^a}$ to denote a set of training sentences for the auxiliary tasks. Note that the sentences in $\mathcal{D}^a$ can be from the sentences in $\mathcal{D}^s$ and $\mathcal{D}^t$, but they can also be from other unlabeled sentences.

3.3 Sentence Embeddings for Domain Adaptation

With the two auxiliary tasks, we can learn a neural network model in a standard way to produce sentence embeddings that work well for the auxiliary tasks. Specifically, we still use Θ' to denote the parameters of the neural network that produces the sentence embeddings (and $f_{\Theta'}$ the corresponding transformation function), and we use $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$ to denote the parameters of two softmax classifiers for the two auxiliary tasks, respectively. Using cross-entropy loss, we can learn Θ' by minimizing the following loss function:

$$\begin{aligned} & J(\Theta', \boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \\ = & -\sum_{(\boldsymbol{x},u,v)\in\mathcal{D}^a} \Big(\log p(u|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^+) \\ & + \log p(v|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^-) \Big), \end{aligned}$$

where $p(y|\boldsymbol{z};\boldsymbol{\beta})$ is the probability of label y given vector $\boldsymbol{z}$ and parameter $\boldsymbol{\beta}$ under softmax regression.

With the learned Θ', we can derive a sentence embedding $\boldsymbol{z}'$ from any sentence. Although we could simply use this embedding $\boldsymbol{z}'$ for sentiment classification through another softmax classifier, this may not be ideal because $\boldsymbol{z}'$ is transformed from $g(\boldsymbol{x})$, which has the domain-independent sentiment words removed. Similar to SCL and some other previous work, we concatenate the embedding vector $\boldsymbol{z}'$ with the standard embedding vector $\boldsymbol{z}$ for the final classification.

3.4 Joint Learning

Although we can learn Θ' using $\mathcal{D}^a$ as a first step, here we also explore a joint learning setting. In this setting, Θ' is learned together with the neural network model used for the end task, i.e., sentiment classification. This way, the learning of Θ' depends not only on $\mathcal{D}^a$ but also on $\mathcal{D}^s$, i.e., the sentiment-labeled training data from the source domain.

Specifically, we use Θ to denote the parameters for a neural network that takes the original sentence $\boldsymbol{x}$ and transforms it to a sentence embedding (and f_Θ the corresponding transformation function). We use γ to denote the parameters of a softmax classifier that operates on the concatenated sentence embedding $\boldsymbol{z} \oplus \boldsymbol{z}'$ for sentiment classification. With joint learning, we try to minimize the following loss function:

$$\begin{aligned}
& J(\Theta, \Theta', \boldsymbol{\gamma}, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-) \\
= & -\sum_{(\boldsymbol{x},y)\in\mathcal{D}^s} \Big(\log p(y|f_\Theta(\boldsymbol{x}) \oplus f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\gamma}) \Big) \\
& -\sum_{(\boldsymbol{x},u,v)\in\mathcal{D}^a} \Big(\log p(u|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^+) \\
& + \log p(v|f_{\Theta'}(g(\boldsymbol{x})); \boldsymbol{\beta}^-) \Big).
\end{aligned}$$

We can see that this loss function contains two parts. The first part is the cross-entropy loss based on the true sentiment labels of the sentences in $\mathcal{D}^s$. The second part is the loss based on the auxiliary tasks and the data $\mathcal{D}^a$, which are derived from unlabeled sentences.

Finally, to make a prediction on a sentence, we use the learned Θ and Θ' to derive a sentence embedding $f_\Theta(\boldsymbol{x}) \oplus f_{\Theta'}(g(\boldsymbol{x}))$, and then use the softmax classifier parameterized by the learned $\boldsymbol{\gamma}$ to make the final prediction.

3.5 Implementation Details

In this section we explain some of the model details.

Pivot Word Selection

Recall that the two auxiliary tasks depend on two domain-independent sentiment word lists, i.e., pivot word lists. Different from Blitzer et al. (2007), we employ weighted log-likelihood ratio (WLLR) to select the most positive and negative words in both domains as pivots. The reason is that in our preliminary experiments we observe that mutual information (used by Blitzer et al. (2007)) is biased towards low frequency words. Some high frequency words including *good* and *great* are scored low. In comparison, WLLR does not have this issue. The same observation was also reported previously by Li et al. (2009).

More specifically, we first tokenize the sentences in $\mathcal{D}^s$ and $\mathcal{D}^t$ and perform part-of-speech tagging using the NLTK toolkit. Next, we extract only adjectives, adverbs and verbs with a frequency of at least 3 in the source domain and at least 3 in the target domain. We also remove negation words such as *not* and stop words using a stop word list. We then measure each remaining candidate word's relevance to the positive and the negative classes based on $\mathcal{D}^s$ by computing the following scores:

$$r(w, y) \quad = \quad \tilde{p}(w|y) \log \frac{\tilde{p}(w|y)}{\tilde{p}(w|\bar{y})},$$

where w is a word, $y \in \{+, -\}$ is a sentiment label, $\bar{y}$ is the opposite label of y, and $\tilde{p}(w|y)$ is the empirical probability of observing w in sentences labeled with y. We can then rank the candidate words in decreasing order of $r(w, +)$ and $r(w, -)$. Finally, we select the top 25% from each ranked list as the final lists of pivot words for the positive and the negative sentiments. Some manual inspection shows that most of these words are indeed domain-independent sentiment words.

Neural Network Model

Our framework is general and potentially we can use any neural network model to transform an input sentence to a sentence embedding vector. In this paper, we adopt a CNN-based approach because it has been shown to work well for sentiment classification. Specifically, each word (including the token *UNK*) is represented by a word embedding vector. Let $\mathbf{W} \in \mathbb{R}^{d\times V}$ denote the lookup table for words, where each column is a d-dimensional embedding vector for a word type. Two separate CNNs are used to process $\boldsymbol{x}$ and $g(\boldsymbol{x})$, and their mechanisms are the same. For a word x_i in each CNN, the embedding vectors inside a window of size n centered at i are concatenated into a new vector, which we refer to as $\mathbf{e}_i \in \mathbb{R}^{nd}$. A convolution operation is then performed by applying a filter $\mathbf{F} \in \mathbb{R}^{h\times nd}$ on $\mathbf{e}_i$ to produce a hidden vector $\mathbf{h}_i = m(\mathbf{F}\mathbf{e}_i + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^h$ is a bias vector and m is an element-wise non-linear transformation function. Note that we pad the original sequence in front and at the back to ensure that at each position i we have n vectors to be combined into $\mathbf{h}_i$. After the convolution operation is applied to the whole sequence, we obtain $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots]$, and we apply a max-over-time pooling operator to take the maximum value of each row of $\mathbf{H}$ to obtain an overall hidden vector, i.e., $\boldsymbol{z}$ for $\boldsymbol{x}$ and $\boldsymbol{z}'$ for $g(\boldsymbol{x})$.

It is worth noting that the two neural networks corresponding to f_Θ and $f_{\Theta'}$ share the same word embedding lookup table. This lookup table is initialized with word embeddings from *word2vec*[1] and

[1]`https://code.google.com/p/word2vec/`

is updated during our learning process. Note that the token *UNK* is initialized as a zero vector and never updated.

3.6 Differences from SCL

Although our method is inspired by SCL, there are a number of major differences: (1) Our method is based on neural network models with continuous, dense feature representations and non-linear transformation functions. SCL is based on discrete, sparse feature vectors and linear transformations. (2) Although our pivot word selection is similar to that of SCL, in the end we only use two auxiliary tasks while SCL uses much more pivot prediction tasks. (3) We can directly learn the transformation function f'_Θ that produces the hidden representation, while SCL relies on SVD to learn the projection function. (4) We perform joint learning of the auxiliary tasks and the end task, i.e., sentiment classification, while SCL performs the learning in a sequential manner.

4 Experiments

4.1 Data Sets and Experiment Settings

Data Set	# Sentences	# Words
Movie1(MV1)	10662	18765
Movie2(MV2)	9613	16186
Camera(CR)	3770	5340
Laptop(LT)	1907	2837
Restaurant(RT)	1572	2930

Table 1: Statistics of our data sets.

To evaluate our proposed method, we conduct experiments using five benchmark data sets. The data sets are summarized in Table 1. *Movie1*[2] and *Movie2*[3] are movie reviews labeled by Pang and Lee (2005) and Socher et al. (2013), respectively. *Camera*[4] are reviews of digital products such as MP3 players and cameras (Hu and Liu, 2004). *Laptop* and *Restaurant*[5] are laptop and restaurant reviews taken from SemEval 2015 Task 12.

We consider 18 pairs of data sets where the two data sets come from different domains.[6] For neural network-based methods, we randomly pick 200 sentences from the target domain as the development set for parameter tuning, and the rest of the data from the target domain as the test data.

4.2 Baselines and Hyperparameters

We consider the following baselines:

Naive is a non-domain-adaptive baseline based on bag-of-word representations.

SCL is our implementation of the Structural Correspondence Learning method. We set the number of induced features K to 100 and rescale factor α = 5, and we use 1000 pivot words based on our preliminary experiments.

mDA is our implementation of marginalized Denoising Auto-encoders (Chen et al., 2012), one of the state-of-the-art domain adaptation methods, which learns a shared hidden representation by reconstructing pivot features from corrupted inputs. Following Yang and Eisenstein (2014), we employ the efficient and effective structured dropout noise strategy without any parameter. The top 500 features are chosen as pivots based on our preliminary experiments.

NaiveNN is a non-domain-adaptive baseline based on CNN, as described in Section 3.5.

Aux-NN is a simple combination of our auxiliary tasks with NaiveNN, which treats the derived label of two auxiliary tasks as two features and then appends them to the hidden representation learned from CNN, followed by a softmax classifier.

SCL-NN is a naive combination of SCL with NaiveNN, which appends the induced representation from **SCL** to the hidden representation learned from CNN, followed by a softmax classifier.

mDA-NN is similar to **SCL-NN** but uses the hidden representation derived from **mDA**.

Sequential is our proposed method without joint learning, which first learns Θ' based on $\mathcal{D}^a$ and then learns Θ and γ based on $\mathcal{D}^s$ with fixed Θ'.

Joint is our proposed joint learning method, that is, we jointly learn Θ and Θ'.

[2]`https://www.cs.cornell.edu/people/pabo/movie-review-data/`

[3]`http://nlp.stanford.edu/sentiment/`

[4]`http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`

[5]Note that the original data set is for aspect-level sentiment analysis. We remove sentences with opposite polarities towards different aspects, and use the consistent polarity as the sentence-level sentiment of each remaining sentence.

[6]Because *Movie1* and *Movie2* come from the same domain, we do not take this pair.

Task		Method									
Source	Target	Naive	Naive++	SCL++	mDA++	NaiveNN	Aux-NN	SCL-NN	mDA-NN	Sequential	Joint
MV1	*LT*	0.656	0.739	0.742	0.742	0.773	0.779	0.776	0.780	0.774	**0.804***
MV1	*RT*	0.625	0.742	0.750	0.761	0.802	0.794	0.817	0.819	0.814	**0.825***
MV1	*CR*	0.609	0.684	0.688	0.688	0.721	0.717	0.734	0.730	0.717	**0.747***
MV2	*LT*	0.699	0.760	0.765	0.772	0.805	0.811	0.800	0.811	0.808	**0.827***
MV2	*RT*	0.696	0.761	0.768	0.778	0.813	0.819	0.824	0.825	0.833	**0.840***
MV2	*CR*	0.644	0.697	0.705	0.706	0.738	0.732	0.736	0.756	0.745	**0.768***
CR	*LT*	0.780	0.791	0.802	0.806	0.848	0.848	0.846	0.850	0.856	**0.858***
CR	*RT*	0.746	0.784	0.782	0.789	0.827	0.835	0.841	0.839	0.835	**0.844***
CR	*MV1*	0.593	0.597	0.612	0.612	0.685	0.689	0.689	0.692	0.687	**0.696***
CR	*MV2*	0.609	0.629	0.644	0.640	0.735	0.726	0.734	0.731	0.735	**0.736**
LT	*RT*	0.736	0.781	0.800	0.810	0.819	0.820	0.823	**0.852**	0.841	0.840
LT	*MV1*	0.574	0.601	0.612	0.630	**0.711**	0.703	0.702	0.709	0.705	0.707
LT	*MV2*	0.588	0.632	0.645	0.663	0.742	0.745	0.739	**0.747**	0.746	**0.747**
LT	*CR*	0.736	0.762	0.768	0.780	0.791	0.796	0.803	**0.819**	0.803	0.817
RT	*LT*	0.732	0.777	0.777	0.799	0.817	0.822	0.831	0.826	0.828	**0.834***
RT	*MV1*	0.580	0.604	0.618	0.643	0.721	0.726	0.724	**0.734**	0.722	0.724
RT	*MV2*	0.605	0.630	0.633	0.664	0.761	0.762	0.756	**0.772**	0.757	0.765
RT	*CR*	0.689	0.708	0.704	0.732	0.764	0.772	0.759	0.774	0.772	**0.779***
Average		0.661	0.704	0.712	0.723	0.770	0.772	0.774	0.781	0.777	**0.787**

Table 2: Comparison of classification accuracies of different methods. * indicates that our joint method is significantly better than NaiveNN, Aux-NN, SCL-NN and mDA-NN with $p < 0.05$ based on McNemar's paired significance test.

For **Naive**, **SCL** and **mDA**, we use LibLinear[7] to train linear classifiers and use its default hyperparameters. In all the tasks, we use unigrams and bigrams with a frequency of at least 4 as features for classification. For the word embeddings, we set the dimension d to 300. For CNN, we set the window size to 3. Also, the size of the hidden representations $\boldsymbol{z}$ and $\boldsymbol{z}'$ is set to 100. Following Kim (2014), the non-linear activation function in CNN is Relu, the mini-batch size is 50, the dropout rate α equals 0.5, and the hyperparameter for the l_2 norms is set to be 3. For **Naive**, **SCL** and **mDA**, we do not use the 200 sentences in the development set for tuning parameters. Hence, for fair comparison, we also include settings where the 200 sentences are added to the training set. We denote these settings by **++**.

4.3 Results

In Table 2, we report the results of all the methods. It is easy to see that the performance of **Naive** is very limited, and the incorporation of 200 reviews in the development set (**Naive++**) brings in 4.3% of improvement on average. **SCL++** and **mDA++** can further improve the average accuracy respectively by 0.8% and 1.9%, which verifies the usefulness of these two domain adaptation methods. However, we can easily see that the performance of these domain adaptation methods based on discrete, bag-of-word representations is even much lower than the non-domain-adaptive method on continuous representations (**NaiveNN**). This confirms that it is useful to develop domain adaptation methods based on embedding vectors and neural network models.

Moreover, we can find that the performance of simply appending two features from auxiliary tasks to **NaiveNN** (i.e., **Aux-NN**) is quite close to that of **NaiveNN** on most data set pairs, which shows that it is not ideal for domain adaptation. In addition, although the shared hidden representations derived from SCL and mDA are based on traditional bag-of-word representations, **SCL-NN** and **mDA-NN** can still improve the performance of **NaiveNN** on most data set pairs, which indicates that the derived shared hidden representations by SCL and by mDA can generalize better across domains and are generally useful for domain adaptation.

Finally, it is easy to see that our method with joint learning outperforms **SCL-NN** on almost all the data set pairs. And in comparison with **mDA-NN**, our method with joint learning can also outper-

[7] `http://www.csie.ntu.edu.tw/cjlin/liblinear/`

Task		Method		
Source	Target	NaiveNN	mDA-NN	Joint
MV1	*LT*	0.802	0.799	**0.816***
MV1	*RT*	0.816	0.820	**0.838***
MV1	*CR*	0.744	0.757	**0.767***
MV2	*LT*	0.823	0.830	**0.839***
MV2	*RT*	0.837	0.829	**0.850***
MV2	*CR*	0.753	0.769	**0.773***
CR	*LT*	0.853	0.863	**0.870***
CR	*RT*	0.840	**0.856**	0.851
CR	*MV1*	0.699	0.701	**0.704***
CR	*MV2*	**0.745**	0.741	**0.745**
LT	*RT*	0.839	**0.849**	**0.849**
LT	*MV1*	0.714	0.710	**0.720***
LT	*MV2*	0.759	**0.767**	0.766
LT	*CR*	0.803	**0.815**	0.814
RT	*LT*	0.825	0.839	**0.841**
RT	*MV1*	0.724	**0.737**	0.732
RT	*MV2*	0.762	**0.771**	0.768
RT	*CR*	0.773	0.777	**0.783***
Average		0.784	0.790	**0.796**

Table 3: Comparison of our method **Joint** with **NaiveNN** and **mDA-NN** in a setting where some labeled target data is used.

form it on most data set pairs, especially when the size of the labeled data in the source domain is relatively large. Furthermore, we can easily observe that for our method, joint learning generally works better than sequential learning. All these observations show the advantage of our joint learning method.

In Table 3, we also show the comparison between **mDA-NN** and our model under a setting some labeled target data is used. Specifically, we randomly select 100 sentences from the development set and mix them with the training set. We can observe that our method **Joint** outperforms **NaiveNN** and **mDA-NN** by 1.2% and 0.6%, respectively, which further confirms the effectiveness of our model. But, in comparison with the setting where no target data is available, the average improvement of our method over **NaiveNN** is relatively small.

Hence, to give a deeper analysis, we further show the comparison of **Joint** and **NaiveNN** with respect to the number of labeled target data in Figure 2. Note that for space limitation, we only present the results on $MV2 \rightarrow RT$ and $MV2 \rightarrow CR$. Similar trends have been observed on other data set pairs. As we can see from Figure 2, the difference between the performance of **NaiveNN** and that of **Joint** gradually decreases with the increase of the number of labeled

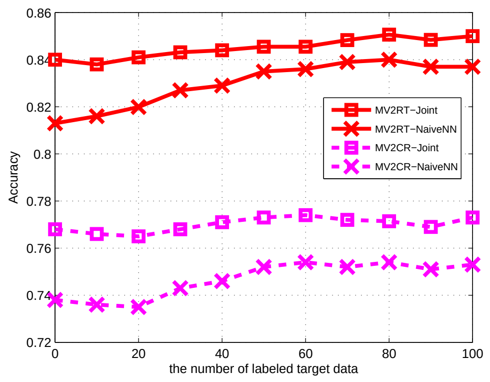


Figure 2: The influence of the number of labeled target data.

target data. This indicates that our joint model is much more effective when no or small number of labeled target data is available.

4.4 Case Study

To obtain a better understanding of our method, we conduct a case study where the source is *CR* and the target is *RT*.

For each sentiment polarity, we try to extract the most useful trigrams for the final predictions. Recall that our CNN models use a window size of 3, which corresponds to trigrams. By tracing the final prediction scores back through the neural network, we are able to locate the trigrams which have contributed the most through max-pooling. In Table 4, we present the most useful trigrams of each polarity extracted by **NaiveNN** and by the two components of our sequential and joint method. **Sequential-original** and **Joint-original** refer to the CNN corresponding to f_Θ while **Sequential-auxiliary** and **Joint-auxiliary** refer to the CNN corresponding to $f_{\Theta'}$, which is related to the auxiliary tasks.

In Table 4, we can easily observe that for **NaiveNN**, the most important trigrams are domain-independent, which contain some general sentiment words like *good*, *great* and *disappointing*. For our sequential model, the most important trigrams captured by **Sequential-original** are similar to **NaiveNN**, but due to the removal of the pivot words in each sentence, the most important trigrams extracted by **Sequential-auxiliary** are domain-specific, including target-specific sentiment words like *oily*, *friendly* and target-specific aspect words like *flavor*, *atmosphere*. But since aspect words are irrelevant to our sentiment classification

Method	Negative Sentiment	Positive Sentiment
NaiveNN	disappointing_*_*, **disgusting_*_***, it_is_not, slow_*_*, *_too_bad, *_*_terrible, **place_is_not**, unpleasant_experience_*, **would_not_go**, *_the_only	*_*_great, good_*_*,*_*_best, *_i_love, was_very_good,*_*_excellent, wonderful_*_*, *_*_amazing, *_*_nice
Sequential-original	**disgusting_*_***, disappointing_*_*, *_*_terrible expensive_*_*, it_is_not, unpleasant_experience_*, slow_*_*, *_too_bad, probably_would_not, awful_*_*	*_*_great, good_*_*, *_*_best, *_i_love, *_highly_recommended, *_*_excellent, wonderful_*_*, is_amazing_*, is_the_perfect
Sequential-auxiliary	**disgusting_*_***, **never_go_back**, money_*_*, **rude_*_***, flavor_*_*, *_this_place, **oily_*_***, prices_*_*, **inedible_!_***, **this_place_survives**	**delicious_*_***, **friendly_*_***, food_*_*, food_is_UNK, *_highly_UNK, **fresh_*_***, atmosphere_*_*, *_i_highly, nyc_*_*
Joint-original	**disgusting_*_***, **soggy_*_***, disappointing_*_*, *_too_bad, *_would_never, it_is_not, **rude_*_***, *_*_terrible, **place_is_not**, disappointment_*_*	*_*_great, good_*_*, *_*_best, *_i_love, *_*_amazing, **delicious_*_***, **back_*_***, *_i_highly, of_my_favorite
Joint-auxiliary	**soggy_*_***, **disgusting_*_***, **rude_*_***, disappointment_*_*, **not_go_back**, **was_not_fresh**, prices_*_*, **inedible_!_***, **oily_*_***, overpriced_*_*	**delicious_*_***, **go_back_***, **is_always_fresh**, **friendly_*_***, **to_die_for**, also_very_UNK, of_my_favorite, food_*_*, *_i_highly, **delicious_!_***

Table 4: Comparison of the most useful trigrams chosen by our method and by NaiveNN on *CR* → *RT*. Here * denotes a "padding", which we added at the beginning and the end of each sentence. The domain-specific sentiment words are in **bold**.

task, it might bring in some noise and affect the performance of our sequential model. In contrast to **Sequential-auxiliary**, **Joint-auxiliary** is jointly learnt with the sentiment classification task, and it is easy to see that most of its extracted trigrams are target-specific sentiment words. Also, for **Joint-original**, since we share the word embeddings of two components and do not remove any pivot, it is intuitive to see that the extracted trigrams contain both domain-independent and domain-specific sentiment words. These observations agree with our motivations behind the model.

Finally, we also sample several sentences from the test dataset, i.e., *RT*, to get a deeper insight of our joint model. Although **NaiveNN** and **Sequential** correctly predict sentiments of the following two sentences:

1. "*I've also been amazed at all the new additions in the past few years: A new Jazz Bar, the most fantastic Dining Garden, the Best Thin Crust Pizzas, and now a Lasagna Menu which is* ***to die for****!*"
2. "*The have a great cocktail with Citrus Vodka and lemon and lime juice and mint leaves that is* ***to die for****!*"

Both of them give wrong predictions on another three sentences containing ***to die for***:

3. "*Try their chef's specials– they are* ***to die for***."
4. "*Their tuna tartar appetizer is* ***to die for***."
5. "*It's* ***to die for****!*".

However, since ***to die for*** co-occurs with some general sentiment words like *fantastic*, *best* and *great* in previous two sentences, our joint model can implicitly learn that ***to die for*** is highly correlated with the positive sentiment via our auxiliary tasks, and ultimately make correct predictions for the latter three sentences. This further indicates that our joint model can identify more domain-specific sentiment words in comparison with **NaiveNN** and **Sequential**, and therefore improve the performance.

5 Conclusions

We presented a domain adaptation method for sentiment classification based on sentence embeddings. Our method induces a sentence embedding that works well across domains, based on two auxiliary tasks. We also jointly learn the cross-domain sentence embedding and the sentiment classifier. Experiment results show that our proposed joint method can outperform several highly competitive domain adaptation methods on 18 source-target pairs using five benchmark data sets. Moreover, further analysis confirmed that our method is able to pick up domain-specific sentiment words.

Acknowledgment

This research is supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

References

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

John Blitzer, Mark Dredze, and Fernando Pereira. 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*, pages 440–447, Prague, Czech Republic, June. Association for Computational Linguistics.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 132–141. Association for Computational Linguistics.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 730–740, Beijing, China, July. Association for Computational Linguistics.

Danushka Bollegala, Tingting Mu, and John Goulermas. 2016. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge & Data Engineering*, 6(2):398–410.

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

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics.

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

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014a. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Baltimore, Maryland, June. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014b. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

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

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, October.

Shoushan Li, Rui Xia, Chengqing Zong, and Chu-Ren Huang. 2009. A framework of feature selection methods for text categorization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 692–700. Association for Computational Linguistics.

Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 410–419. Association for Computational Linguistics.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In

Proceedings of the 19th international conference on World wide web, pages 751–760. ACM.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Natalia Ponomareva and Mike Thelwall. 2012. Do neighbours help?: an exploration of graph-based algorithms for cross-domain sentiment classification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 655–665. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal, September. Association for Computational Linguistics.

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

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 538–544.

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

Jianfei Yu and Jing Jiang. 2015. A hassle-free unsupervised domain adaptation method using instance similarity features. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 168–173, Beijing, China, July. Association for Computational Linguistics.