12-2018

Feature-based transfer learning In natural language processing

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

FEATURE-BASED TRANSFER LEARNING IN NATURAL LANGUAGE PROCESSING

JIANFEI YU

SINGAPORE MANAGEMENT UNIVERSITY
2018

Feature-Based Transfer Learning In Natural Language Processing

by

Jianfei YU

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Jing JIANG (Supervisor / Chair)
Associate Professor of Information Systems
Singapore Management University

Hady Wirawan LAUW
Associate Professor of Information Systems
Singapore Management University

Feida ZHU
Associate Professor of Information Systems
Singapore Management University

Sinno Jialin PAN
Nanyang Assistant Professor of Computer Engineering
Nanyang Technological University

Singapore Management University
2018

I hereby declare that this PhD dissertation is my original work and it has been written by me in its entirety.
I have duly acknowledged all the sources of information which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree in any university previously.

Jianfei Yu
5 Dec 2018

Feature-Based Transfer Learning In Natural Language Processing

Jianfei YU

Abstract

In the past few decades, supervised machine learning approach is one of the most important methodologies in the Natural Language Processing (NLP) community. Although various kinds of supervised learning methods have been proposed to obtain the state-of-the-art performance across most NLP tasks, the bottleneck of them lies in the heavy reliance on the large amount of manually annotated data, which is not always available in our desired target domain/task. To alleviate the data sparsity issue in the target domain/task, an attractive solution is to find sufficient labeled data from a related source domain/task. However, for most NLP applications, due to the discrepancy between the distributions of the two domains/tasks, directly training any supervised models only based on labeled data in the source domain/task usually results in poor performance in the target domain/task. Therefore, it is necessary to develop effective transfer learning techniques to leverage rich annotations in the source domain/task to improve the model performance in the target domain/task.

There are generally two settings of transfer learning. We use *supervised transfer learning* to refer to the setting when a small amount of labeled target data is available during training, and when no such data is available we call it *unsupervised transfer learning*. In this thesis, we focus on proposing novel transfer learning methods for different NLP tasks in both settings, with the goal of inducing an invariant latent feature space across domains or tasks, where the knowledge gained from the source domain/task can be easily adapted to the target domain/task.

In the *unsupervised transfer learning* setting, we first propose a simple yet effective domain adaptation method by deriving shared representations with instance similarity features, which can be generally applied for different NLP tasks, and em-

pirical evaluation on several NLP tasks shows that our method has indistinguishable or even better performance than a widely used domain adaptation method. Furthermore, we target at a specific NLP task, i.e., sentiment classification, and propose a neural domain adaptation framework, which performs joint learning of the actual sentiment classification task and several manually designed domain-independent auxiliary tasks to produce shared representations across domains. Extensive experiments on both sentence-level and document-level sentiment classification demonstrate that our proposed domain adaptation framework can achieve promising results.

In the *supervised transfer learning* setting, we first propose a neural domain adaptation approach for retrieval-based question answering systems by simultaneously learning shared feature representations and modelling inter-domain and intra-domain relationships in a unified model, followed by conducting both intrinsic and extrinsic evaluation to demonstrate the efficiency and effectiveness of our method. Moreover, we attempt to improve multi-label emotion classification with the help of sentiment classification by proposing a dual attention transfer network, where a shared feature space is employed to capture the general sentiment words, and another task-specific space is employed to capture the specific emotion words. Experimental results show that our method is able to outperform several highly competitive transfer learning methods.

Although the transfer learning methods proposed in this thesis are originally designed for natural language processing tasks, most of them can be potentially applied to classification tasks in the other research communities such as computer vision and speech processing.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recent years have witnessed the explosive growth of digitized textual data, particularly user-generated contents on the Web, such as customer reviews, microblogs and forum posts. To automatically analyze and mine the useful knowledge from the great volume of online textual data, various kinds of natural language processing (NLP) tasks including sentiment analysis, information extraction (IE), paraphrase identification (PI) and question answering (QA) have received continuous attention from both the academia and industry.

For different NLP tasks, supervised machine learning technologies, especially deep learning approaches, have been extensively studied in the last decade. However, to obtain a well-trained model for each specific task, these supervised approaches typically require hundreds of thousands of annotated data points, such as relation extraction [91], natural language inference [17], machine comprehension [90] and question answering [16]. But in real scenarios, such large-scale data sets may be only available in certain well-studied domains/tasks, while for our desired target domain/task, we may have little or even no annotated data.

To train a robust model for our target domain/task, a direct solution is to manually construct a well-annotated data set by domain/task experts. Nevertheless, such

an annotating process is generally time-consuming and costly due to its labor intensive nature. By contrast, a much more attractive solution with minimal human guidance is to borrow labeled data or useful knowledge from a related resource-rich source domain/task to help train a robust machine learning model for the target domain/task, which is known as *transfer learning* or *domain adaptation*. Motivated by this, my research centers around developing effective *transfer learning* techniques to enable knowledge transfer from *source* to *target* for different NLP problems, so that we can reach highly competitive performance in the resource-poor target domain/task.

The key challenge of *transfer learning* is that directly applying the model trained in a resource-rich source domain/task in our desired resource-poor domain/task may lead to a significant performance drop. Take the widely used Stanford's natural language inference (SNLI) data set [17] as an example. The original SNLI data set is collected from Flickr30k image captions [127] with over 570K labeled sentence pairs, but if we directly apply the model trained on SNLI to make predictions for sentence pairs in the *Fiction* domain of their extended data set MNLI [112], the performance will drop by 41.6 point percentages (80.6% to 47.1% in accuracy by their CBOW model). The major reason behind this phenomenon is that the vocabularies used in the source and the target domains differ a lot, which causes the discrepancy between the two domains' underlying distributions. As a remedy, if we are able to find a shared feature space across domains where the source knowledge can be easily transferred to the target, it will largely reduce the distribution gap between domains and significantly improve the domain adaptation performance. This is the motivation for me to explore how to effectively induce a shared feature space between the source and the target domains/tasks (a.k.a feature-based transfer learning) under two commonly used *transfer learning* settings: (1) *unsupervised* transfer learning: no labeled samples from the target domain are available while a large amount of labeled samples from the source domain are available; and (2) *supervised* transfer learning: a small amount of labeled samples from the target do-

main/task and a large amount of labeled samples from the source domain/task are both available.

In the literature, a myriad of *transfer learning* approaches have been proposed for different NLP tasks in the two settings respectively.

Unsupervised Transfer Learning[1]: Most existing unsupervised domain adaptation methods either leverage the well-known dimensionality reduction technique [12, 14, 76, 77] or apply the stacked denoising auto-encoder (SDA) [38, 21, 63, 120, 140] to automatically learn the domain-independent feature representations. However, these approaches generally suffer from the following two limitations. **First**, most of them are computationally intensive and cannot easily scale to large-scale data sets. Take the well-known Structural Correspondence Learning (SCL) method [12] as an example. Its first step is to identify a set of domain-independent features and construct thousands of binary classifiers by using remaining features to predict the presence of each domain-independent feature. In order to induce a shared representation for each instance, it further performs singular value decomposition (SVD) on the learnt weights to obtain a low dimensional mapping matrix. While it has been widely used in many NLP tasks, the training of thousands of binary classifiers in the first step will inevitably result in high computational costs. Moreover, in practice, it is not easy to perform dimensionality reduction on a large-scale data set, which limits its applications in real scenarios. **Second**, most existing unsupervised domain adaptation approaches are on basis of traditional discrete features and linear classifiers, and it is less clear how to perform transfer learning based on the continuous word embeddings and advanced neural network models. Although a number of studies have successfully applied various kinds of neural network models to different NLP tasks such as sentiment classification [95, 50, 103], information extraction [95, 56, 130, 67], natural language inference [17, 70, 51] and question answering [102, 125] to achieve promising performance, most of them only focus on the standard in-domain settings while failing

[1]For this setting, *transfer learning* is the same as *domain adaptation* in this dissertation.

to pay attention to the challenging domain adaptation settings. Therefore, it would be interesting to propose effective unsupervised domain adaptation approaches on top of these neural network (NN) models. Motivated by these two limitations, we aim to (1) propose a novel unsupervised domain adaptation method, which incurs relatively low computational costs and is also suitable for large-scale data sets; (2) and propose novel NN-based approaches that can generally derive robust feature representations against domain shift for different NLP tasks.

Supervised Transfer Learning: Since this setting assumes to have labeled data in both the source and the target domains/tasks, most existing supervised transfer learning methods essentially adopt different multi-task learning strategies to improve the model performance in the target domain/task. One line of work employs *fine-tuning* strategy, where they first use the source labeled data to pre-train the model parameters, followed by fine-tuning the parameters based on the target labeled data. Another line of work leverages the labeled data from source and target to perform *joint learning*, where the goal is to automatically learn the shared feature representations. Although these approaches have been shown to bring significant improvements to the target domain/task, simple multi-task learning strategies without any explicit constraints cannot guarantee that their shared feature space is really dominated by the shared features across domains/tasks as we expect. Inspired by such an intuition, we aim to explicitly incorporate several constraints into existing multi-task learning frameworks, so that the shared feature space and the domain/-task specific feature space are respectively dominated by the shared features and the domain/task specific features.

1.2 Methodology

In the motivation section, we have pointed out that existing transfer learning solutions still suffer from several major limitations. Therefore, in this section, we will give a brief introduction of our proposed approaches for addressing these lim-

itations respectively. Note that the first two approaches belong to unsupervised transfer learning, whereas the latter two approaches belong to supervised transfer learning.

1.2.1 A Hassle-free Unsupervised Domain Adaptation Method with Instance Similarity Features

To solve the first challenge that existing solutions to unsupervised domain adaptation are either computationally expensive or unscalable in large-scale applications, we propose a simple yet effective unsupervised domain adaptation method.

Specifically, our method first picks up a subset of unlabeled instances from the target domain and then normalizes them to form a set of *exemplar vectors*. For each instance in both the source and the target domains, these exemplar vectors are further utilized to produce a new feature vector, where the value of each dimension is computed by the instance's similarity with each exemplar vector. With the new feature vector, we finally concatenate them together with its original feature vector to represent each source or target instance, followed by performing the main task on this new feature space. Essentially, instead of relying on time-consuming dimensionality reduction or SDA to derive the shared feature space, our method creates a new feature space simply based on similarities between instances and exemplar vectors. This allows our method to enjoy faster computation and robust scalability.

1.2.2 An Unsupervised Neural Domain Adaptation Method with Auxiliary Tasks for Sentiment Classification

As mentioned in the motivation section, most existing unsupervised domain adaptation methods are based on traditional linear classifiers, and how to effectively extend them to neural network-based models is still unclear to us. On the other hand, since the neural network architectures designed for different NLP tasks usually differ a lot, it is next to impossible to propose a general neural domain adaptation method

for all NLP tasks. Therefore, we first target at sentiment classification, a classic but important task in NLP, and propose an unsupervised neural domain adaptation framework for this task.

In particular, we first devise two alternative auxiliary tasks inspired by SCL method [10]: one is to predict the occurrence of both positive and negative domain-independent words; the other is to predict the sum of sentiment scores of domain-independent words. With the two carefully designed auxiliary tasks, we further propose a general domain adaptation framework, and then leverage two existing neural network architectures to jointly learn the auxiliary task and our main sentiment classification task under the framework in a multi-task learning fashion. Since the auxiliary tasks are general across domains, we expect the hidden representations corresponding to the auxiliary task to be robust in both domains, and hence improve the cross-domain performance.

1.2.3 A Supervised Neural Domain Adaptation Method for Retrieval-based Question Answering Systems

To encourage the shared (or domain-specific) feature space to capture more shared (or domain-specific) features across domains, we propose a new transfer learning framework by modelling the intra-domain and the inter-domain relationships, and apply it to another specific task, Retrieval-based Question Answering.

Specifically, we propose a new supervised transfer learning method by explicitly modeling the domain relationships via a covariance matrix, which imposes a regularization term on the weights of the output layer to uncover both the inter-domain and intra-domain relationships. Furthermore, to make the shared representation more invariant across domains, we follow some recent work on adversarial networks [36, 64] and introduce an adversarial loss on the shared feature space in our method. To optimize the whole framework, we employ an alternating optimization method, which allows us to jointly learn the parameters in the neural networks

and the domain relationships in an end-to-end fashion.

1.2.4 A Neural Task Adaptation Method for Improving Multi-label Emotion Classification via Sentiment Classification

Finally, motivated by the same intuition as above, we propose another novel supervised transfer learning architecture to guarantee the difference between the shared feature space and the task-specific feature space, and apply it to improve the performance of the resource-poor multi-label emotion classification task with the help of the resource-rich sentiment classification task.

In particular, we first propose a simple shared-private model, where we employ a shared LSTM layer to extract shared sentiment features for both sentiment and emotion classification tasks, and a target-specific LSTM layer to extract specific emotion features that are only sensitive to our emotion classification task. To enforce the orthogonality of the two feature spaces, we further introduce a dual attention mechanism, which feeds the attention weights in one feature space as extra inputs to compute those in the other feature space, and explicitly minimizes the similarity between the two sets of attention weights.

1.3 Our Contributions

In this dissertation, we study feature-based transfer learning and its applications in different natural language processing tasks, aiming to improve the model performance in the resource-scarce domain/task with the help of rich annotations in some related domains/tasks. The main contributions of this dissertation can be summarized as follows:

Firstly, we propose a simple yet effective unsupervised domain adaptation method [128], which reduces high computational costs and can easily scale to large-scale applications. We theoretically show that our method is able to assign

appropriate weights to target-specific features, which co-occur with useful domain-independent features. Our extensive evaluations on four NLP tasks show that our method can generally outperform several baselines including the widely used SCL.

Secondly, we develop an unsupervised neural network-based domain adaptation framework together with two novel auxiliary tasks for sentiment classification, and respectively apply them to sentence-level sentiment classification [129] and document-level sentiment classification [131] based on two state-of-the-art neural network models. We conduct a series of experiments to examine the effectiveness of our proposed framework.

Thirdly, we propose a supervised neural domain adaptation method for retrieval-based question answering systems, which can explicitly capture the inter-domain and intra-domain relationships. Both intrinsic and extrinsic evaluation demonstrate that our approach method can efficiently and effectively improve the model performance in the resource-poor target domain [133].

Finally, we design a dual attention transfer learning network, which is expected to capture the shared features and the target-specific features in two different feature spaces. Experiments on both English and Chinese data sets show that the proposed architecture is able to bring significant improvements to the resource-poor emotion classification task [132].

1.4 Organization

As illustrated by Figure 1.1, the remaining part in this dissertation is organized as follows. We first conduct a comprehensive review of related work about transfer learning in Chapter 2. In Chapter 3, we present our unsupervised domain adaptation algorithm based on instance similarity features. Next, Chapter 4 introduces our neural unsupervised domain adaptation framework followed by two specific neural architectures for both sentence-level and document-level sentiment classification. Furthermore, we detail our supervised transfer learning framework by explicitly

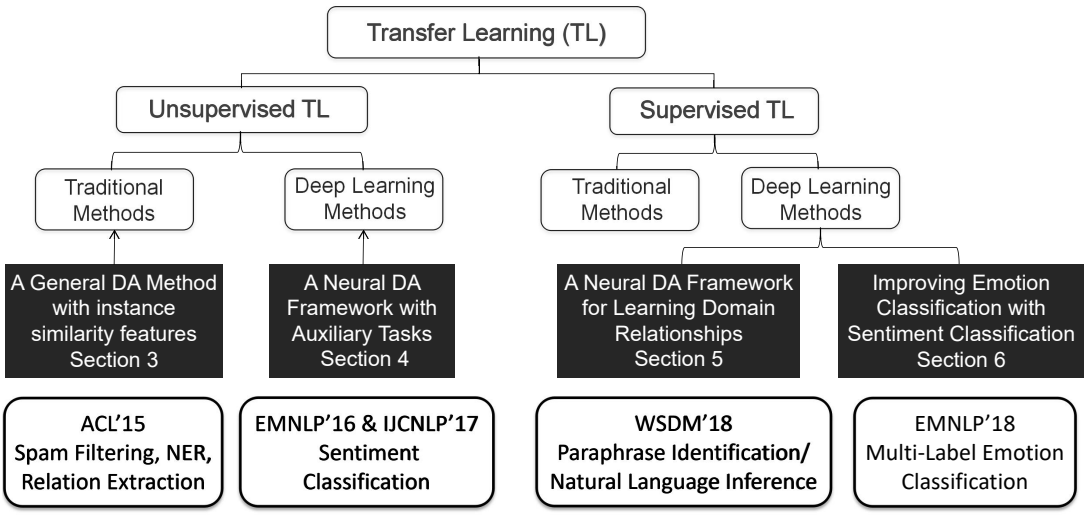

Figure 1.1: The Organization of the Topics in this Thesis.

modelling domain relationships in Chapter 5. Moreover, Chapter 6 describes our dual attention-based supervised transfer learning network. Finally, we conclude this dissertation and discuss our future work in Chapter 7.

Chapter 2

Literature Review

In this chapter, we will give a comprehensive literature review over existing studies related to this dissertation.

Before reviewing related work, let us first introduce the necessary notation. Without losing generality, we assume a binary classification problem where each input is represented as a feature vector $\boldsymbol{x}$ from an input vector space $\mathcal{X}$ and the output is a label $y \in \{0, 1\}$. Our goal is to learn an optimal mapping function f^* to map the input variable X to the output variable Y. This assumption is general because many NLP tasks such as text categorization, named entity recognition (NER), relation extraction (RE), natural language inference (NLI) and answer selection (AS) can be cast into classification problems and our discussion below can be easily extended to multi-class classification settings.

Next, we will introduce existing transfer learning methods based on two common transfer learning settings: *unsupervised* transfer learning and *supervised* transfer learning, where the key difference is that the latter setting assumes to have a small number of labeled target data but the former one assumes that no labeled target data are available.

2.1 Unsupervised Transfer Learning

Unsupervised Transfer Learning, also referred to as *unsupervised domain adaptation* in this dissertation, assumes that we only have enough labeled data from a source domain but may also have a large amount of unlabeled data from both the source and the target domains [12]. To facilitate our discussion, let us use $D^{\mathrm{s}} = \{(\boldsymbol{x}_i^{\mathrm{s}}, y_i^{\mathrm{s}})\}_{i=1}^{N}$ and $D^{\mathrm{s,u}} = \{\boldsymbol{x}_j^{\mathrm{s,u}}\}_{j=1}^{N^{\mathrm{s,u}}}$ to represent both the labeled and the unlabeled instances from a source domain, where $\mathcal{X}_{\mathrm{s}}$ is defined to be the subspace of $\mathcal{X}$ spanned by all source domain input vectors. We further use $D^{\mathrm{t,u}} = \{\boldsymbol{x}_j^{\mathrm{t,u}}\}_{j=1}^{N^{\mathrm{t,u}}}$ to denote the unlabeled instances from a target domain, and $\mathcal{X}_{\mathrm{t}}$ to denote the subspace of all target domain input vectors. Besides, we also use $D^{\mathrm{t}} = \{(\boldsymbol{x}_i^{\mathrm{t}}, y_i^{\mathrm{t}})\}_{i=1}^{N^{\mathrm{t}}}$ to denote the target instances in the test set, and assume that the output label y^{s} and y^{t} are from the same label set, i.e., the target task is the same as the source task. The goal of unsupervised domain adaptation is to utilize D^{s}, $D^{\mathrm{s,u}}$ and $D^{\mathrm{t,u}}$ to train a robust classifier, which can perform well on D^{t}.

To achieve this goal, let us first try to understand why a well-trained classifier on D^{s} cannot perform well on D^{t}. As analyzed in [47], the essential reason behind the performance drop in this domain adaptation setting is that the joint distribution of source domain $\mathcal{P}_s(X, Y)$ is different from that of target domain $\mathcal{P}_t(X, Y)$, i.e., $\mathcal{P}_s(X, Y) \neq \mathcal{P}_t(X, Y)$. Since no labeled target instance is available in this setting, it is very challenging to directly estimate $\mathcal{P}_t(X, Y)$. Following [47], we can slightly simplify the problem by decomposing $\mathcal{P}(X, Y)$ into $\mathcal{P}(X)\mathcal{P}(Y|X)$, and explain the difference between $\mathcal{P}_s(X, Y)$ and $\mathcal{P}_t(X, Y)$ from two perspectives:

Feature Mismatch: In this case, we assume $\mathcal{P}_s(X) = \mathcal{P}_t(X)$ so that the difference comes from $\mathcal{P}_s(Y|X) \neq \mathcal{P}_t(Y|X)$, which is essentially caused by the difference between two domains' input subspaces, i.e., $\mathcal{X}_s \neq \mathcal{X}_t$. Hence, in this thesis, we call it as feature mismatch.

Instance Mismatch: In this case, we assume $\mathcal{P}_s(Y|X) = \mathcal{P}_t(Y|X)$ while the marginal distribution of two domains are different from each other, i.e., $\mathcal{P}_s(X) \neq$

$\mathcal{P}_t(X)$. The reason for $\mathcal{P}_s(X) \neq \mathcal{P}_t(X)$ is intuitive since some representative instances in the source domain may be not so important in the target domain, and vice verse. Therefore, we call it as instance mismatch hereafter.

These two kinds of assumptions naturally lead us to two kinds of solutions to domain adaptation problems: feature-based adaptation and instance-based adaptation. In fact, most existing domain adaptation studies belong to these two branches [78, 65], and thus we will review previous works from the two perspectives respectively. Note that there is another line of work called self-labelling, which focuses on labelling the unlabeled target instances $D^{\text{t,u}}$ by using the initial model trained on the labeled source instances D^{s}, and then iteratively adding the instances with high confidence into the training set via self-training [3, 26]. Since these studies are quite similar to semi-supervised learning and out of the scope of this thesis, we will not review them here.

2.1.1 Feature-Based Approaches

Feature-based domain adaptation assumes that the core reason leading to the domain difference is $\mathcal{X}_s \neq \mathcal{X}_t$. Therefore, the key solution to such a problem is to project $\mathcal{X}_s$ and $\mathcal{X}_t$ into a shared feature space $\mathcal{X}_c$, so that all instances in the source and the target domains can be represented in the same feature space $\mathcal{X}_c$.

To solve this problem, one possible solution is to make use of $D^{\text{s,u}}$ and $D^{\text{t,u}}$ to learn a transformation function $\phi(\boldsymbol{x})$ in an unsupervised learning manner. After the transformation, each source input vector $\boldsymbol{x}^{\text{s}}$ and target input vector $\boldsymbol{x}^{\text{t}}$ can be further represented by $\phi(\boldsymbol{x}^{\text{s}})$ and $\phi(\boldsymbol{x}^{\text{t}})$, which are in the same space $\mathcal{X}_c$.

In the last decade, a large amount of domain adaptation approaches have been proposed to learn the transformation function. Most of them are based on traditional discrete one-hot representations and linear classifiers, while more recent explorations are based on continuous word vectors and non-linear deep learning classifiers. Accordingly, we classify them into two groups in the following: traditional

methods and deep learning methods.

Traditional Methods

For traditional feature-based domain adaptation methods, most of them can be generally categorized into the following three clusters.

One line of work focuses on leveraging the co-occurrences between domain-specific and domain-independent features followed by dimensionality reduction to learn $\phi(\boldsymbol{x})$. Among them, the most representative work is the Structural Correspondence Learning (SCL) method [12]. SCL first chooses a set of pivot features based on either term frequency or feature selection methods like mutual information, denoted by $\mathcal{D}$. Then, it constructs $|\mathcal{D}|$ binary classifiers to predict the presence of each pivot feature based on all the non-pivot features, denoted by $\mathcal{V} - \mathcal{D}$, where $\mathcal{V}$ refers to the whole feature set. Next, the learnt weights of all the classifiers are arranged into a matrix $W \in \mathbb{R}^{|D|\times|\mathcal{V}-\mathcal{D}|}$, and the top K left singular vectors are selected via Singular Value Decomposition (SVD) to form a dimension-reduced matrix $\theta \in \mathbb{R}^{K\times|\mathcal{V}-\mathcal{D}|}$. Finally, each source and target instance is multiplied with θ to obtain a new K-dimensional feature vector. That is, the transformation function is $\phi(\boldsymbol{x}) = \theta\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}-\mathcal{D}|}$ means that each instance is only represented by non-pivot features. Later, on the basis of the learning framework of SCL, Pan et al. [76] proposed to construct a bipartite graph by connecting the pivot and non-pivot features through their co-occurrence statistics, followed by using spectral clustering to learn the dimension-reduced matrix θ as in SCL. Almost at the same time, Pan et al. proposed another dimensionality reduction-based approach named transfer component analysis (TCA) to achieve the same goal [77]. Since these works only focused on inducing a shared feature space without bearing the target task in mind, the learnt feature space $\mathcal{X}_c$ may not be sensitive and critical for the target task. Based on such an intuition, a more recent study [14] proposed to induce a task-sensitive embedding space across domains by encoding the label of target tasks into the learning phase.

In addition, denoising auto-encoders have been extensively applied to learn

$\phi(\boldsymbol{x})$, 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 [38], and taking one of the hidden layer out as $\phi(\boldsymbol{x})$ for each instance $\boldsymbol{x}$. Later, to reduce the computational cost and scale to high-dimensional features, Chen et al. [21] proposed to analytically marginalize over the noise during SDA training, and only reconstruct the domain-independent features. Based on the two works mentioned above, Yang et al. [120] further showed that their proposed structured dropout noise strategy can dramatically improve the efficiency without sacrificing the accuracy. Recently, Zhou et al. [138] explored the possibility of transferring between both domains by constraining the distribution consistency with a linear reconstruction.

Apart from these two lines of research, there is another line of work which aims to directly derive target domain-specific features to address the feature mismatch problem, but most of them are designed for sentiment analysis, which limits their generality. In particular, Bollegala et al. [15] first generated a sentiment-sensitive thesaurus including words with similar sentiment polarities from both the source and the target domains, and then appended related target specific features to the original feature space based on the thesaurus. Li et al. [58] proposed a bootstrapping approach by exploiting the relationship between topic and sentiment words across domains to extract lexicons in the target domain.

However, the first line of work is generally based on various dimensionality reduction techniques, and therefore unscalable in large-scale data sets; the second line of work requires high computational cost due to the training of multi-layer auto-encoders; and the third line of work is only limited to the specific cross-domain sentiment classification task. Different from all existing approaches reviewed above, we propose a simple domain adaptation method based on instance similarity features in Chapter 3, which is fast, scalable and general for any NLP tasks.

Deep Learning Methods

Although different deep learning methods have been well applied in standard in-domain setting, the research on developing deep learning-based domain adaptation methods is relatively scarce.

One line of work is on top of the recent popular word embedding methods (e.g., the skip-gram method in [66]), which aims to learn a cross-domain representation for each feature based on predicting its neighboring features. In particular, Bollegala et al. [13] and Yang et el. [121] first selected a set of pivot features as in SCL, and then proposed to use pivots to predict their co-occurring non-pivots, with a constraint that the learnt feature embeddings for the same pivot feature in both domains are close to each other. Different from this line of work which targets at learning general word or feature representations, we proposed to derive general sentence embeddings via a multi-layer neural network as in Chapter 4. But their learnt word embeddings can be used to initialize our neural network architecture.

In addition, another line of work [36, 62] aims to utilize the recent hot adversarial neural network to learn a shared sentence embedding for each sentence in both domains, which is closest to our work in Chapter 4. Specifically, they incorporate a domain classifier on top of the last hidden layer, and introduce an adversarial loss to make the domain classifier can not distinguish between the source and the target domains on the last hidden layer. Since the domain classifier is not necessarily related to the main sentiment classification task, it may result in the learnt shared representations are task insensitive. Nevertheless, our proposed methods in Chapter 4 for learning shared representations are different from them. In particular, we manually designed two auxiliary tasks, which are closely related to the main task and ensure that the induced embeddings are task-sensitive.

2.1.2 Instance-Based Approaches

Instance-based domain adaptation assumes that $\mathcal{P}_s(X) \neq \mathcal{P}_t(X)$ is the intrinsic reason under the discrepancy between two domains' joint distributions. Therefore, the key challenge for addressing this problem is how to transform the marginal distribution of source domain $\mathcal{P}_s(X)$ so as to approximate the marginal distribution of target domain $\mathcal{P}_t(X)$, which is also known as the *covariate shift* problem [93].

To tackle this challenge, one possible solution is to mine from the source labeled data to find the instances that are similar to the marginal distribution of the target domain, which is usually called as *Instance Selection* and has been applied in different NLP tasks. Specifically, Axelrod et al. [4] proposed to select target-domain-similar instances from the source domain for machine translation by ranking source sentences with respect to target sentences based on simple cross-entropy. Later, Xia et al. [114] utilized Positive and unlabeled learning (a.k.a, PU Learning) for cross-domain sentiment classification, which essentially assigns an in-target-domain probability to each source instance via PU Learning, and only uses instances with high in-target-domain probability for model training.

Instead of *Instance Selection*, a more elegant way to solve the *covariate shift* problem is usually referred to as *Instance Weighting*, i.e., assigning a new weight to each source instance so as to approximate the target domain distribution $\mathcal{P}_t(X)$. Let us use $\hat{\mathcal{P}}_t(X) = \mathbf{w}(X)\mathcal{P}_s(X)$ to denote the approximated target distribution. The key challenge of this line of work is how to estimate the instance weight or instance importance $\mathbf{w}(X) = \frac{\mathcal{P}_t(X)}{\mathcal{P}_s(X)}$, which is also known as density ration estimation (DRE) problem. In machine learning community, the DRE problem has been well studied. Shimodaira et at. [93] and Huang et at. [45] respectively proposed to use kernel density estimation and kernel mean matching to directly produce the importance weight $\mathbf{w}(\boldsymbol{x}_i^{\mathrm{s}})$ for each source instance $\boldsymbol{x}_i^{\mathrm{s}}$. Later, due to the two methods' limited scalability in cross validation framework, Sugiyama et al. [97] and Kanamori et al. [48] further proposed to respectively minimize the Kullback-Leibler (KL) divergence

and least square distance between the approximated distribution $\hat{\mathcal{P}}_t(X)$ and the real target distribution $\mathcal{P}_t(X)$ by estimating $\mathbf{w}(X)$ with a linear function in a Gaussian kernel space. However, all these instance weighting methods are limited to low-dimensional feature space, which is hard to be applied into many NLP tasks with high-dimensional feature space. To extend these instance weighting approaches to text classification, Xia et al. [116] first used a feature selection method to choose an effective subset of original features for domain adaptation, and then proposed to minimize the statistical distance between $\hat{\mathcal{P}}_t(X)$ and $\mathcal{P}_t(X)$ following the instance weighting framework in [97].

2.2 Supervised Transfer Learning

Different from *unsupervised* transfer learning, *supervised* transfer learning assumes that we not only have sufficient labeled data from a source domain/task but also have a little labeled data from a target domain/task [78]. To be consistent with previous notations, we use $D^{\mathrm{s}} = \{(\boldsymbol{x}_i^{\mathrm{s}}, y_i^{\mathrm{s}})\}_{i=1}^{N}$ and $D^{\mathrm{t,l}} = \{(\boldsymbol{x}_j^{\mathrm{t,l}}, y_j^{\mathrm{t,l}})\}_{j=1}^{N^{\mathrm{t,l}}}$ to represent them respectively. Due to the availability of the target data, in this setting, the output labels y^{s} and $y^{\mathrm{t,l}}$ are unnecessarily from the same label set, i.e., the source task and the target task can be different. Besides, we may also have some unlabeled data from both domains, which are still denoted by $D^{\mathrm{s,u}}$ and $D^{\mathrm{t,u}}$.

Similar to unsupervised transfer learning, we will review previous work according to feature-based and instance-based approaches under this setting.

2.2.1 Feature-Based Approaches

Recall that the core idea behind feature-based transfer learning methods is to find a latent shared feature space $\mathcal{X}_c$, where the model trained on D^{s} can be well transferred to the target domain/task.

Traditional Methods

To achieve this goal, Daumé III et al. [28] first proposed an easy domain adaptation (EA) method by simply duplicating the original feature space $\mathcal{X}_s \cup \mathcal{X}_t$ by three times to form the new feature space $\mathcal{X}_c$, followed by training a single model on such an augmented feature space based on both D^{s} and $D^{\mathrm{t,l}}$. Later, they extended EA to a semi-supervised version by incorporating the unlabeled target data $D^{\mathrm{t,u}}$, which is used as a regularizer to make the model trained on D^{s} and the model trained on $D^{\mathrm{t,l}}$ become closer. More recently, Xiao et al. [117] designed a kernel matching method to learn the shared feature space $\mathcal{X}_c$ by simultaneously learning a classifier on D^{s} and mapping all the target instances $D^{\mathrm{t,l}} \cup D^{\mathrm{t,u}}$ to a subset of D^{s}.

Deep Learning Methods

With the recent advances of deep learning, different neural network (NN)-based supervised transfer learning frameworks have been proposed for image processing [126] and speech recognition [107] as well as NLP applications [70, 122, 64]. Since we have both labeled source instances D^{s} and labeled target instances $D^{\mathrm{t,l}}$, a simple but widely used framework is referred to as *fine-tuning*, which first uses the parameters of the well-trained models on D^{s} to initialize the model parameters of the target domain/task, and then fine tune the parameters based on $D^{\mathrm{t,l}}$ [126, 70]. Another line of work can be referred to as *multi-task feature learning*, which bears the same intuition behind the traditional feature-based methods as mentioned above. Among this line of work, one typical framework [70, 122] is to simply use a shared NN to learn a shared feature space $\mathcal{X}_c$ but with two different output layers respectively corresponding to D^{s} and $D^{\mathrm{t,l}}$, which is usually referred to as fully-shared framework. More recently, another representative framework [64] is proposed to employ a shared NN and two domain/task specific NNs to respectively derive a shared feature space $\mathcal{X}_c$ and two domain/task specific feature space, which is referred to as specific-shared framework for short.

Comparing traditional and deep learning methods for supervised transfer learning, we are more interested in the NN-based specific-shared framework since it has been shown to achieve the state-of-the-art performance on several NLP tasks like cross-domain text classification. In particular, motivated by the observation that the specific-shared framework fails to consider the domain relationship, we focus on proposing a novel model by jointly learning the shared feature representations and domain relationships in a unified framework. Moreover, inspired by the recent success of applying adversarial networks into unsupervised domain adaptation [36, 101] and multi-task learning [64], we also aim to incorporate the adversarial training into our model in order to learn a more robust shared feature space across domains/tasks.

2.2.2 Instance-Based Approaches

Since the research on instance-based supervised transfer learning is relatively limited, especially for NLP tasks, we will briefly go through some representative work in this subsection.

As a classic work for *Instance Selection*, Jiang et al. [47] proposed an intuitive approach to select a subset of D^{s} that are similar to instances in the target domain, where they first use the model trained on $D^{t,l}$ to predict all instances in D^{s}, and then filter out the top K wrongly predicted instances. By doing so, the remaining instances in D^{s} would have a higher similarity to target instances, and therefore are expected to be useful for predicting the label of the test set D^{t}. On the other hand, for *Instance Weighting*, Dai et al. [27] extended the well-known Adaboost method to domain adaptation problems by assigning relatively larger weights to those mistakenly predicted target instances in $D^{t,l}$ and relatively smaller weights to D^{s} in each iteration. Because the instance weighting strategy proposed in [27] did not bear the density ratio problem in mind, Bickel et al. [9] further devised a generative model to estimate the parameters in both the density ratio and final

discriminative classifier by leveraging a joint MAP hypothesis.

However, all the instance-based transfer learning methods reviewed in Section 2.1.2 and Section 2.2.2 are still based on traditional methods with discrete features and linear classifiers. It will be interesting to extend some representative methods to deep learning methods by verifying their effectiveness, and we would like to leave it to our future work.

Part I

Unsupervised Transfer Learning

Chapter 3

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

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

3.1 Introduction

Domain adaptation aims to use labeled data from a source domain to help build a system for a target domain, possibly with a small amount of labeled data from the target domain. The problem arises when the target domain has a different data distribution from the source domain, which is often the case. In NLP, domain adaptation

has been well studied in recent years. Existing work has proposed both techniques designed for specific NLP tasks [20, 29, 119, 85, 44, 73, 74] and general approaches applicable to different tasks [12, 28, 47, 33, 104]. With the recent trend of applying deep learning in NLP, deep learning-based domain adaptation methods [38, 21, 120] have also been adopted for NLP tasks [121].

As introduced in Chapter 2, there are generally two settings of unsupervised transfer learning (or domain adaptation). *Supervised domain adaptation* refers to the setting when a small amount of labeled target data is available, and *unsupervised domain adaptation* refers to the setting when no such data is available during training. In both settings, unlabeled target domain data can be used.

Although many domain adaptation methods have been proposed, for practitioners who wish to avoid implementing or tuning sophisticated or computationally expensive methods due to either lack of enough machine learning background or limited resources, simple approaches are often more attractive. A notable example is the frustratingly easy domain adaptation method proposed by [28], which simply augments the feature space by duplicating features in a clever way. However, this method is only suitable for supervised domain adaptation. A later semi-supervised version of this easy adaptation method uses unlabeled data from the target domain [30], but it still requires some labeled data from the target domain. In this chapter, we propose a general unsupervised domain adaptation method that is almost equally hassle-free but does not use any labeled target data.

Our method uses a set of unlabeled target instances to induce a new feature space, which is then combined with the original feature space. We explain analytically why the new feature space may help domain adaptation. Using a few different NLP tasks, we then empirically show that our method can indeed learn a better classifier for the target domain than a few baselines. In particular, our method performs consistently better than or competitively with Structural Correspondence Learning (SCL) [12], a well-known unsupervised domain adaptation method in NLP. Furthermore, compared with SCL and other advanced methods such as the marginalized

structured dropout method [120] and a recent feature embedding method [121], our method is much easier to implement.

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

3.2 Adaptation with Similarity Features

Before presenting our method, we first recall the necessary notation as introduced in Chapter 2. To facilitate our discussion, we consider a binary classification problem, where each input is represented as a feature vector $\boldsymbol{x}$ and $y \in \{0, 1\}$ is the true label of $\boldsymbol{x}$. Besides, we have a set of labeled instances from a source domain, denoted by $D^{\text{s}} = \{(\boldsymbol{x}_i^{\text{s}}, y_i^{\text{s}})\}_{i=1}^{N}$. We also have a set of unlabeled instances from a target domain, denoted by $D^{\text{t}} = \{\boldsymbol{x}_j^{\text{t}}\}_{j=1}^{M}$. We assume a general setting of learning a linear classifier, which is essentially a weight vector $\boldsymbol{w}$ such that $\boldsymbol{x}$ is labeled as 1 if $\boldsymbol{w}^\top \boldsymbol{x} \geq 0$.[1]

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

3.2.1 The Method

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

$$g(\boldsymbol{x}) = [s(\boldsymbol{x}, \boldsymbol{e}^{(1)}), \ldots, s(\boldsymbol{x}, \boldsymbol{e}^{(K)})]^\top, \tag{3.1}$$

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

where $\top$ indicates transpose and $s(\boldsymbol{x}, \boldsymbol{x}')$ is a similarity function between $\boldsymbol{x}$ and $\boldsymbol{x}'$. In our work we use dot product as s.[2] Once each labeled source domain instance is transformed into a K-dimensional vector by Equation 3.1, we can append this vector to the original feature vector of the source instance and use the combined feature vectors of all labeled source instances to train a classifier. To apply this classifier to the target domain, each target instance also needs to add this K-dimensional induced feature vector.

It is worth noting that the exemplar vectors are randomly chosen from the available target instances and no special trick is needed. Overall, the method is fairly easy to implement, and yet as we will see in Section 4.5, it performs surprisingly well. We also want to point out that our instance similarity features bear strong similarity to what was proposed by [98], but their work addresses a completely different problem and we developed our method independently of their work.

3.2.2 Justification

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

Learning in the Target Subspace

Blitzer et al. [11] pointed out that the hope of unsupervised domain adaptation is to "couple" the learning of weights for target-specific features with that of common features. We show our induced feature representation is exactly doing this.

First, we review the claim by [11]. We note that although the input vector space $\mathcal{X}$ is typically high-dimensional for NLP tasks, the actual space where input vectors lie can have a lower dimension because of the strong feature dependence we observe with NLP tasks. For example, binary features defined from the same fea-

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

ture template such as the previous word are mutually exclusive. Furthermore, the actual low-dimensional spaces for the source and the target domains are usually different because of domain-specific features and distributional difference between the domains. Recall that in Chapter 2, we define subspace $\mathcal{X}_{\text{s}}$ to be the (lowest dimensional) subspace of $\mathcal{X}$ spanned by all source domain input vectors. Similarly, a subspace $\mathcal{X}_{\text{t}}$ can be defined. Define $\mathcal{X}_{\text{s,t}} = \mathcal{X}_{\text{s}} \bigcap \mathcal{X}_{\text{t}}$, the shared subspace between the two domains. Define $\mathcal{X}_{\text{s},\perp}$ to be the subspace that is orthogonal to $\mathcal{X}_{\text{s,t}}$ but together with $\mathcal{X}_{\text{s,t}}$ spans $\mathcal{X}_{\text{s}}$, that is, $\mathcal{X}_{\text{s},\perp} + \mathcal{X}_{\text{s,t}} = \mathcal{X}_{\text{s}}$. Similarly we can define $\mathcal{X}_{\perp,\text{t}}$. Essentially $\mathcal{X}_{\text{s,t}}$, $\mathcal{X}_{\text{s},\perp}$ and $\mathcal{X}_{\perp,\text{t}}$ are the shared subspace and the domain-specific subspaces, and they are mutually orthogonal.

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

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

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

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

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

Blitzer et al. [11] argues that with unlabeled target instances, we can hope to "couple" the learning of $\boldsymbol{w}_{\perp,\text{t}}$ with that of $\boldsymbol{w}_{\text{s,t}}$. We show that if we use only our induced feature representation without appending it to the original feature vector, we can achieve this. We first define a matrix $M_{\mathcal{E}}$ whose column vectors are the exemplar vectors from $\mathcal{E}$. Then $g(\boldsymbol{x})$ can be rewritten as $M_{\mathcal{E}}^\top \boldsymbol{x}$. Let $\boldsymbol{w}'$ denote a

linear classifier learned from the transformed labeled data. $\boldsymbol{w}'$ makes prediction based on $\boldsymbol{w}'^{\top} M_{\mathcal{E}}^{\top} \boldsymbol{x}$, which is the same as $(M_{\mathcal{E}} \boldsymbol{w}')^{\top} \boldsymbol{x}$. This shows that the learned classifier $\boldsymbol{w}'$ for the induced features is equivalent to a linear classifier $\overline{\boldsymbol{w}} = M_{\mathcal{E}} \boldsymbol{w}'$ for the original features.

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

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

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

In practice, however, we need to append the induced features to the original features to achieve good adaptation results. One may find this counter-intuitive because this results in an expanded instead of restricted hypothesis space. Our explanation is that because of the typical L_2 regularizer used during training, there is an incentive to shift the weight mass to the additional induced features. The need to combine the induced features with original features was also reported in previous domain adaptation work such as SCL [12] and marginalized denoising autoencoders [21].

Reduction of Domain Divergence

Another theory on domain adaptation developed by [8] essentially states that we should use a hypothesis space that can achieve low error on the source domain while at the same time making it hard to separate source and target instances. If we use only our induced features, then $\mathcal{X}_{\mathrm{s},\perp}$ is excluded from the hypothesis space. This is

Features	$\hat{\varepsilon}_s$	domain separation error	$\hat{\varepsilon}_t$
Original	0.000	0.011	0.283
ISF-	0.120	0.129	0.315
ISF	0.006	0.062	**0.254**

Table 3.1: Three errors of different feature representations on a spam filtering task. K is 200 for ISF- and ISF. We expect a low $\hat{\varepsilon}_t$ when $\hat{\varepsilon}_s$ is low and domain separation error is high.

likely to make it harder to distinguish source and target instances. To verify this, in Table 3.1 we show the following errors based on three feature representations: (1) The training error on the source domain ($\hat{\varepsilon}_s$). (2) The classification error when we train a classifier to separate source and target instances. (3) The error on the target domain using the classifier trained from the source domain ($\hat{\varepsilon}_t$). ISF- means only our induced instance similarity features are used while ISF uses combined feature vectors. The results show that ISF achieves relatively low $\hat{\varepsilon}_s$ and increases the domain separation error. These two factors lead to a reduction in $\hat{\varepsilon}_t$.

Difference from EA++

The easy domain adaptation method EA proposed by [28] has later been extended to a semi-supervised version EA++ [30], where unlabeled data from the target domain is also used. Theoretical justifications for both EA and EA++ are given by [54]. Here we briefly discuss how our method is different from EA++ in terms of using unlabeled data. In both EA and EA++, since labeled target data is available, the algorithms still learn two classifiers, one for each domain. In our algorithm, we only learn a single classifier using labeled data from the source domain. In EA++, unlabeled target data is used to construct a regularizer that brings the two classifiers of the two domains closer. Specifically, the regularizer defines a penalty if the source classifier and the target classifier make different predictions on an unlabeled target instance. However, with this regularizer, EA++ does not strictly restrict either the source classifier or the target classifier to lie in the target subspace $\mathcal{X}_t$. In contrast, as we have pointed out above, when only the induced features are used, our method leverages the unlabeled target instances to force the learned classifier to lie in $\mathcal{X}_t$.

3.2.3 Exemplar Vectors Selection

Until now, all the exemplar vectors in our method are randomly chosen from the target instances, and one important assumption we make for these exemplar vectors is that they contain some target-specific features. However, this assumption cannot be guaranteed to be satisfied by simply choosing random instances from the target domain. For example, in the extreme case, if all the chosen instances did not contain any target-specific features but only common features, i.e., $\boldsymbol{x}_{\perp,\mathrm{t}} = \mathbf{0}$, then our method is not able to learn an appropriate weight vector for target-specific features. Therefore, we argue that random selection of exemplar vectors may lead to two potential limitations: (1) it is possible to choose some "poor" exemplar vectors that may only contain common features; and (2) the result of our method might be relatively unstable, since it is highly dependent on the proportion of "poor" exemplar vectors.

To solve these two limitations, we further propose to apply a clustering approach to cluster all the target instances into K clusters, and then treat the K cluster centroids as our exemplar vectors. Specifically, we first employ the well-known K-Means clustering algorithm [49] to perform clustering on the available target instances, where Euclidean distance is used to measure the instance similarity. Next, the resulting K cluster centroids are utilized to form our exemplar vectors:

$$\mathcal{E} = \{\boldsymbol{c}^{(k)}\}_{k=1}^{K},$$

where $\boldsymbol{c}^{(k)}$ refers to the centroid of the k-th cluster. Finally, we still use Eqn. (3.1) to derive the K-dimensional vector for each source and target instance.

3.3 Experiments

In this section we use a few NLP tasks to demonstrate the effectiveness of our method.

3.3.1 Tasks and Data Sets

We consider the following NLP tasks.

Personalized Spam Filtering (Spam): The data set comes from ECML/PKDD 2006 discovery challenge. The goal is to adapt a spam filter trained on a common pool of 4000 labeled emails (i.e., $u*$) to three individual users' personal inboxes, each containing 2500 emails (i.e., *u00*, *u01* and *u02*). We use bag-of-word features for this task, and we report classification accuracy.

Relation Extraction (RE): We use the ACE2005 data where the annotated documents are from several different sources such as broadcast news and conversational telephone speech. We report the F1 scores of identifying the 7 major relation types. We use standard features including entity types, entity head words, contextual words and Part of Speech Tags of head words and contextual words. Note that in this thesis, for simplicity, we remove syntactic features derived from both constituency and dependency parse trees, which have been used in our original paper.

Gene Name Recognition (NER): The data set comes from BioCreAtIvE Task 1B [40]. It contains three sets of Medline abstracts with labeled gene names. Each set corresponds to a single species (fly, mouse or yeast). We consider domain adaptation from one species to another. We use standard NER features including words, POS tags, prefixes/suffixes and contextual features. We report F1 scores for this task.

Sentiment Classification (Sentiment): We employ the benchmark data set, which contains four different domains of Amazon product review, i.e., *Book* (B), *DVD* (D), *Electronics* (E) and *Kitchen* (K), released by [10] for experiments. We use unigrams and bigrams with a frequency of at least 5 as features for this task.

3.3.2 Methods for Comparison

Naive uses the original features.

SCL is our implementation of Structural Correspondence Learning [12]. After tun-

Method	Spam			
	u00	u01	u02	**Average**
Naive	0.680	0.715	0.843	0.746
SCL	0.731	0.753	0.837	0.774
ISF	0.749	0.764	0.892	0.802†
KISF	**0.759**	**0.789**	**0.903**	**0.817**†

Table 3.2: Comparison of performance on **Spam**. For each source-target pair of each task, the performance shown is the average of 5-fold cross validation. We also report the overall average performance for each task. We tested statistical significance only for the overall average performance and found that ISF and KISF were significantly better than both Naive and SCL with $p < 0.05$ (indicated by †) based on the Wilcoxon signed-rank test.

ing the hyperparameters, we set the number of induced features to 25. For pivot features, we follow the setting used by [12] and select the features with a term frequency more than 50 in both domains.

ISF is our method using instance similarity features by randomly selecting target instances as our exemplar vectors. We first transform each training instance to a K-dimensional vector according to Eqn. (3.1) and then append the vector to the original vector.

KISF is a modified version of **ISF**, where we choose the centroids of K clusters from the K-Means algorithm as the exemplar vectors, as described in Section 3.2.3.

For all the three NLP tasks and the methods above that we compare, we train linear classifiers with LibLinear[3] and use its default hyperparameters[4].

3.3.3 Results

In Table 3.2, Table 3.3 and Table 3.4, we show the comparison between our two methods and the baseline approaches on the four tasks respectively. For both ISF and KISF, the parameter K is set to 100 for Spam, 300 for Relation, 1000 for NER and 500 for Sentiment after tuning. As we can see from the table, SCL can improve

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

[4]Note that in our original paper, we employ maximum entropy classification algorithm with L_2 regularization to train a classifier, and use the L-BFGS optimization algorithm to optimize our objective function. Therefore, the results reported in this thesis is slightly different from those in our original paper.

Method	Relation							
	bc→bn	bc→cts	bc→nw	bc→un	bc→wl	bn→bc	bn→cts	bn→nw
Naive	0.412	0.324	0.431	0.344	0.389	0.520	0.425	0.480
SCL	0.430	0.328	0.446	0.348	0.391	0.526	0.402	0.481
ISF	0.432	**0.336**	0.448	0.370	0.404	0.531	0.427	0.485
KISF	**0.447**	0.333	**0.451**	**0.380**	**0.408**	**0.547**	**0.429**	**0.490**
	bn→un	bn→wl	cts→bc	cts→bn	cts→nw	cts→un	cts→wl	nw→bc
Naive	0.443	0.453	0.322	0.307	0.265	0.379	0.327	0.468
SCL	0.454	0.460	0.366	0.316	0.291	0.380	0.333	0.470
ISF	0.453	0.460	0.361	0.327	0.291	0.403	0.347	0.481
KISF	**0.459**	**0.465**	**0.387**	**0.336**	**0.294**	**0.410**	**0.353**	**0.491**
	nw→bn	nw→cts	nw→un	nw→wl	un→bc	un→bn	un→cts	un→nw
Naive	0.406	0.305	0.341	0.427	0.311	0.315	0.374	0.303
SCL	0.420	**0.334**	0.369	0.431	0.346	0.349	**0.388**	0.314
ISF	0.421	0.320	0.358	0.437	0.349	0.347	0.382	0.328
KISF	**0.430**	0.319	**0.369**	**0.437**	**0.374**	**0.361**	0.380	**0.339**
	un→wl	wl→bc	wl→bn	wl→cts	wl→nw	wl→un	**Average**	
Naive	0.317	0.296	0.253	0.213	0.324	0.211	0.356	
SCL	0.304	0.335	**0.303**	0.227	0.353	0.224	0.371	
ISF	0.341	0.328	0.282	0.221	0.347	0.236	0.375	
KISF	**0.356**	**0.340**	0.299	**0.239**	**0.362**	**0.246**	**0.384**†	

Table 3.3: Comparison of performance on **RE**. † indicates that our method KISF was significantly better than both Naive and SCL with $p < 0.05$ based on the Wilcoxon signed-rank test.

the performance in most settings for all three tasks, which confirms the general effectiveness of this method. For our method ISF, we can see that on average it outperforms Naive on all the tasks, and SCL on the first three tasks. This shows that our method is competitive despite its simplicity. Moreover, we can further observe that by using the cluster centroids as exemplar vectors, it can bring significant improvements over our ISF method in all the four tasks. When we zoom into the different source-target domain pairs of the four tasks, we can see that KISF can significantly outperform Naive in almost all the cases, and SCL for most of the cases. Besides, it is also worth pointing out that SCL incurs much more computational cost than ISF and KISF.

Method	NER						
	f→m	f→y	m→f	m→y	y→f	y→m	**Average**
Naive	0.336	0.406	0.131	0.536	0.063	0.347	0.303
SCL	0.334	0.426	0.135	0.537	0.063	0.346	0.307
ISF	0.347	0.424	0.141	0.549	0.064	**0.351**	0.313
KISF	**0.351**	**0.427**	**0.147**	**0.554**	**0.065**	0.349	**0.316**†

Method	Sentiment						
	B→D	B→E	B→K	D→B	D→E	D→K	E→B
Naive	0.776	0.710	0.728	0.751	0.715	0.729	0.692
SCL	**0.794**	0.723	**0.750**	**0.771**	0.727	**0.756**	0.704
ISF	0.785	0.736	0.742	0.745	0.728	0.742	0.700
KISF	0.786	**0.737**	0.743	0.758	**0.734**	0.742	**0.707**

	E→D	E→K	K→B	K→D	K→E	**Average**
Naive	0.687	0.817	0.685	0.698	0.801	0.732
SCL	**0.727**	0.822	**0.730**	**0.736**	0.812	**0.754**
ISF	0.700	0.823	0.703	0.709	0.811	0.744
KISF	0.703	**0.825**	0.705	0.709	**0.815**	0.747

Table 3.4: Comparison of performance on **NER** and **Sentiment**. † indicates that our method KISF was significantly better than both Naive and SCL with $p < 0.05$.

3.3.4 Impact of the Number of Exemplar Vectors

In this subsection, we discuss the impact of the number of exemplar vectors, i.e., the sensitivity of K in our proposed methods ISF and KISF. For simplicity, we only conduct experiments on one subtask of each task, (i.e., *u∗ → u00* for **Spam**, *bc → bn* for **Relation**, *f → y* for **NER** and *B → E* for **Sentiment**). All the results of the four subtasks are illustrated in Figure 3.1.

It is easy to see that when the value of K is relatively small, the performance of both ISF and KISF will generally increase as K increases. But when K increases to a certain value, the performance of both methods tends to be stable, and the performance gain by increasing K becomes smaller and smaller. Take **Relation** in Figure 3.1 for example, we can easily observe that when K reaches 300, the room for further improvements is limited, and therefore the performance becomes stable. Similar trends can be observed from the other tasks Figure 3.1. Generally, we can observe from Figure 3.1 that setting K to 800 to 1000 is able to result in stable

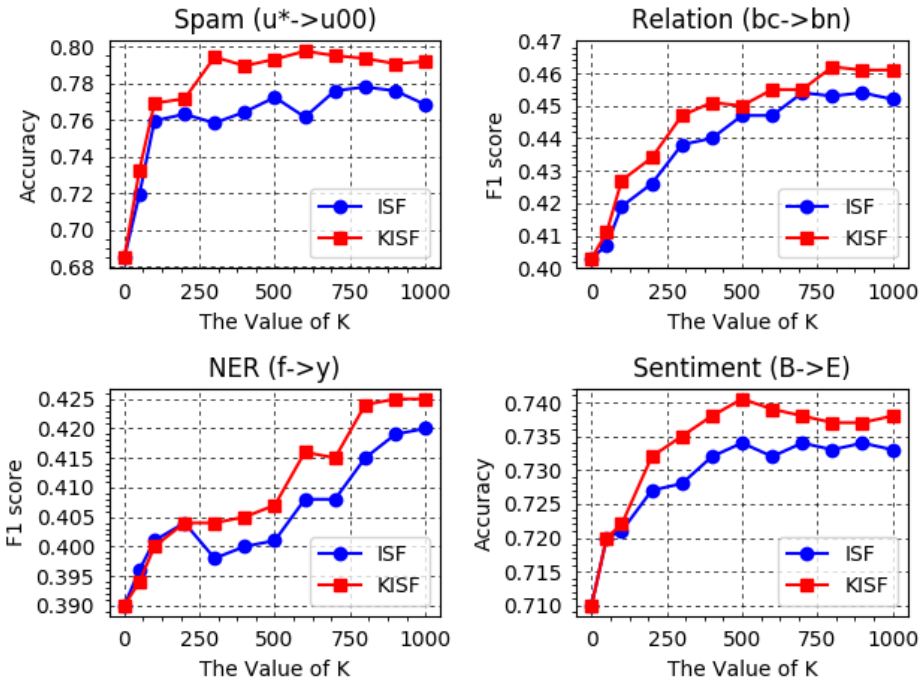

Figure 3.1: The Impact of the Number of Exemplar Vectors.

and highly competitive performance across all the four tasks. But since a lower dimensionality can lead to faster running time, we set K to 100 for **Spam**, 300 for **Relation**, 1000 for **NER** and 500 for **Sentiment** in this paper.

3.3.5 Stability Comparison between ISF and KISF

To verify our analysis in Section 3.2.3, we compare the stability of ISF and KISF in this subsection. Here we follow the above setting by choosing the same subtask of each task for experiments, and run each method with the same setting on a fixed fold of the source-target pair for ten times.

In Figure 3.2, we rank the ten results of each method and show them in an increasing order. It is easy to observe that the gap between the maximum and the minimum value of ISF is much larger than that of KISF, which is in line with our expectation that ISF is relatively unstable. Moreover, we can find that the variance of the ISF method is clearly much higher than that of the KISF method, and this indicates that KISF is more stable than ISF and can generally achieve much better domain adaptation performance.

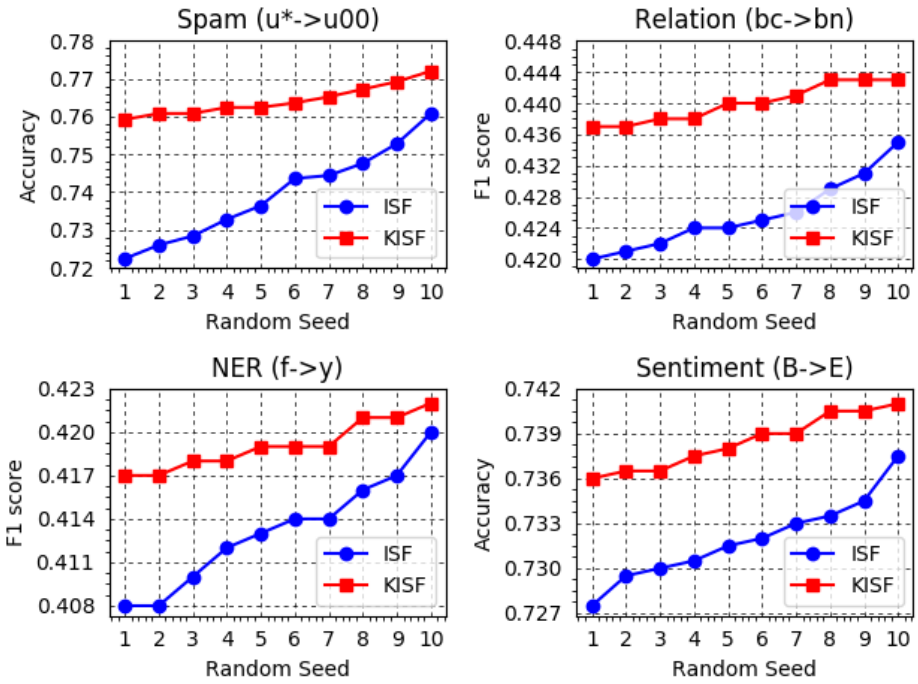

Figure 3.2: The Comparison of the Stability of ISF and KISF on ten runs.

3.4 Discussion

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

Chapter 4

An Unsupervised Neural Domain Adaptation Framework with Auxiliary Tasks for Sentiment Classification

In this chapter, we study cross-domain sentiment classification with neural network architectures. We borrow the idea from Structural Correspondence Learning and carefully design two alternative auxiliary tasks to help induce shared sentence embeddings and document embeddings that supposedly works well across domains for sentiment classification. We further propose a novel domain adaptation framework by incorporating the two auxiliary tasks into two state-of-the-art neural models. Experiment results demonstrate that our proposed framework outperforms several competitive methods on five benchmark data sets of sentence-level sentiment classification and four data sets of document-level sentiment classification.

4.1 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 [81, 79, 43, 23, 71, 115]. One of the big challenges of sentiment classification is how to adapt a sentiment classifier trained on one domain to a different new domain. The reason that this is a big challenge is that sentiments are often expressed with domain-specific words and expressions. For example, in the **Book** domain, expressions such as *an insider's look* and *a must read* are usually positive, but they may not be useful for the **Kitchen** domain. Similarly, words such as *sharp* and *clean*, which are positive in the **Kitchen** domain, can rarely be seen in the **Book** domain. Due to the high cost of obtaining labeled data, it would be very attractive if we can adapt a model trained on a *source domain* to a *target domain*.

A number of different models have been proposed to address this *domain adaptation* problem [10, 76, 15, 86, 114, 14]. Among them, an appealing method is the Structural Correspondence Learning (SCL) method [10], 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 is that these pivot prediction tasks are highly correlated with the original task. In [10], for sentiment classification, the authors 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, various kinds of multi-layer neural network models, including convolutional neural networks (CNNs) [50, 57, 134], recursive neural networks (ReNNs) [95, 32] and recurrent neural networks (RNNs) [99, 123], have been proposed for sentiment classification and achieved good performance. 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 have been some recent studies on neural network-based domain adaptation [38, 21, 120, 138]. Most of these methods center around employing denoising auto-encoders to induce a hidden representation that presumably works well across domains. However, the auto-encoder-based methods are 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 on neural network-based domain adaptation aims to learn a low dimensional representation for each feature in both domains based on predicting its neighboring features [121, 13]. Different from these methods, we aim to directly learn sentence and document embeddings that work well across domains.

In this chapter, we aim to extend the main idea behind SCL to neural network-based models and propose a general domain adaptation framework, where the actual sentiment classification task and our manually designed auxiliary tasks are trained jointly with multi-task learning. Specifically, we first borrow the idea of using pivot prediction tasks from SCL and introduce two kinds of auxiliary tasks. Instead of learning thousands of pivot predictors, our two auxiliary tasks are only based on two binary prediction tasks and one multi-class prediction task, respectively. In addition, different from SCL, which performs singular value decomposition (SVD) on the learned weights of pivot predictors through *linear* transformations, our domain adaptation framework builds upon two well-known neural network models to directly learn a *non-linear* transformation that maps an input to a dense embedding

vector. Moreover, unlike 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.

The main contributions of this chapter can be summarized as follows:

- We develop a neural network-based domain adaptation framework for sentiment classification.
- We further propose two kinds of domain-independent auxiliary tasks and employ two existing neural network models to help us induce robust sentence and document embeddings across domains based on our framework.
- We conduct experiments on a number of different source and target domains for both sentence-level and document-level sentiment classification. The experimental results show that our proposed methods can significantly outperform a number of baselines and are able to achieve comparable or even better results compared with a strong baseline proposed by us.

The rest of this chapter is organized as follows. In Section 4.2, we present an overview of our domain adaptation framework. The two proposed auxiliary tasks are detailed in Section 4.3, and the sentence-level and document-level sentiment adaptation models are given in Section 4.4. In Section 4.5, we report extensive evaluations of our proposed models.

4.2 Domain Adaptation Framework

In this section, we first formally formulate the task and introduce the necessary notation. Next, we present the overview of our domain adaptation framework.

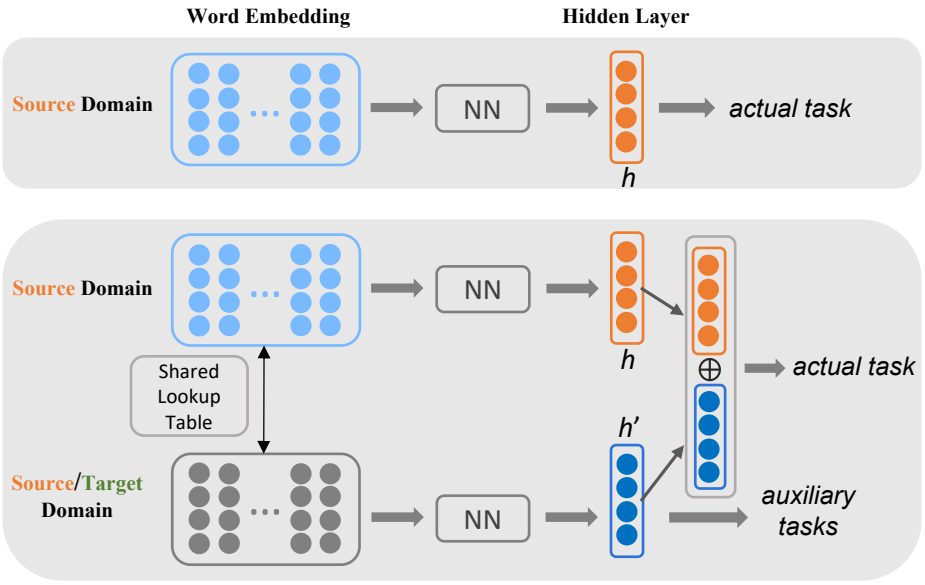

Figure 4.1: Standard Model without Domain Adaptation (top) vs Proposed Domain Adaptation Framework (bottom).

4.2.1 Notation and Task Formulation

Our task is sentiment classification at both the sentence level and the document level. We assume that each input is a piece of text consisting of a sequence of words. To be consistent with the notation introduced in Chpater 2, let $\boldsymbol{x} = (x_1, x_2, \ldots)$ denote an input sentence or document where each $x_i \in \{1, 2, \ldots, |\mathcal{V}|\}$ is a word in the vocabulary and $|\mathcal{V}|$ is the vocabulary size. Let the sentiment label of $\boldsymbol{x}$ be $y \in \{+, -\}$, where $+$ and $-$ denote the positive sentiment and the negative sentiment, respectively.

We consider a cross-domain setting, in which we assume that we have a set of labeled training samples from a source domain, denoted by $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N^s}$. In addition, we have a set of unlabeled samples from a target domain, denoted by $\mathcal{D}^{\text{t,u}} = \{\boldsymbol{x}_i^{\text{t,u}}\}_{i=1}^{N^{\text{t,u}}}$. Our goal is to train a good sentiment classifier using $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t,u}}$ so that the classifier can generally work well in the target domain. To evaluate the trained classifier, we test its performance on a set of labeled samples from the target domain, denoted by $\mathcal{D}^{\text{t,l}} = \{(\boldsymbol{x}_i^{\text{t,l}}, y_i^{\text{t,l}})\}_{i=1}^{N^{\text{t,l}}}$.

4.2.2 Overview of Our Proposed Framework

The core of our domain adaptation framework is to use several domain-independent auxiliary tasks to help induce a cross-domain hidden representation that is useful for both the source and the target domains. The idea of learning cross-domain hidden representations by leveraging auxiliary tasks for domain adaptation is not new [12, 10]. These previous studies essentially follow the multi-task learning framework proposed by [2]. The rationale behind them is that if there are some auxiliary tasks related to the actual prediction task and the labels of the auxiliary tasks can be easily obtained for both the source and the target domains, the induced low-dimensional feature space is a good representation for domain adaptation. Our work follows this line of work and aims to extend the main idea to neural network-based models.

A baseline solution without considering any domain difference is to simply train a classifier using $\mathcal{D}^s$, and we can consider any kind of multi-layer neural networks such as a CNN or an RNN to perform the classification task. We assume that a multi-layer neural network is used to transform each input $\boldsymbol{x}$ into an embedding vector $\mathbf{h}$. Let us use f_Θ to denote the transformation function parameterized by Θ, that is, $\mathbf{h} = f_\Theta(\boldsymbol{x})$. Next, we assume that a linear classifier such as a softmax classifier is learned to map $\mathbf{h}$ to a sentiment label y.

We introduce several 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 auxiliary tasks, we can learn a non-linear transformation function $f_{\Theta'}$ from the unlabeled data and use it to derive an embedding vector $\mathbf{h}'$ from the input $\boldsymbol{x}$, which supposedly works better across domains. Finally we use the source domain's training data to learn a linear classifier on the representation $\mathbf{h} \oplus \mathbf{h}'$, where $\oplus$ is the operator that concatenates two vectors.

Figure 4.1 illustrates the overview of our framework. We can see that in the standard model, the hidden layer $\mathbf{h}$ is learned only through back propagation from the

actual sentiment label y. Since the sentiment labels are only available in the source domain, the learned sentence embeddings and document embeddings may not be sensitive to the target domain. But in our framework, we use the auxiliary hidden layer $\mathbf{h}'$ for predicting not only the actual sentiment labels but also the labels of a carefully designed auxiliary task. Since the auxiliary task is domain-independent, the hidden layer $\mathbf{h}'$ (i.e., sentence embeddings and document embeddings) learned by our method are expected to work well in both domains.

Under this framework, we have the following remaining challenges: (1) designing a good auxiliary task, which is closely related to sentiment classification, and (2) integrating the auxiliary task and the actual sentiment classification task together in a suitable neural network followed by optimizing them together. In the following sections, we will first introduce the two auxiliary tasks we propose and then present our specific neural network models for sentiment classification.

It is worth noting that although we only focus on sentence-level and document-level sentiment classification in this chapter, our framework is general and potentially it can also be used for word-level predictions (i.e., sequence labeling tasks as in [31]), where the key is to design a word-level auxiliary task.

4.3 Auxiliary Tasks for Sentiment Classification

In this section, we introduce our two kinds of auxiliary tasks in detail.

Inspired by SCL [10], we also try to leverage the domain-independent sentiment words (which are referred to as pivots) to set up some auxiliary prediction tasks. The key assumption behind is that if any domain-specific expression occurs frequently with positive pivots like *good* and *wonderful*, this expression is highly likely to be positive in that domain; similarly, for any domain-specific expression which often co-occurs with negative pivots like *bad* and *terrible*, it tends to express a negative sentiment in that domain. For example, in the review "*The laptop is good and goes really fast*," if we know that the sentiment polarity of the pivot word *good*

is positive, the sentiment polarity of the domain-specific word *fast* has a strong possibility to be positive. Based on this assumption, we can first hide all the pivots and design some auxiliary prediction tasks based on these pivots. Then, if we only use the remaining domain-specific words to predict these domain-independent auxiliary tasks, it should be helpful for identifying some important domain-specific sentiment expressions like *goes really fast* in the above example.

The next question is how we should construct the auxiliary tasks based on pivots. In SCL, they proposed to set up thousands of binary classification tasks to predict the occurrence of each pivot. However, if we directly apply their auxiliary tasks into neural network models, it may have two main limitations:

- Thousands of pivot predictors will largely increase the computational cost of optimizing neural network models;
- Since most pivots may only occur in a small amount of samples but not in the other samples of the whole corpus, the training samples for most pivot predictors will be highly imbalanced[1].

Hence, to reduce the computational cost and eliminate the imbalanced sample distribution problem, we consider two alternative solutions: (1) We still use the auxiliary task in SCL but reduce the number of pivots. (2) Instead of predicting the occurrence of each pivot, we propose a new auxiliary task.

4.3.1 Auxiliary Task 1

Since the two limitations mentioned above is essentially caused by the large amount of pivot words, an intuitive way to address the limitations is to first reduce the number of pivots by grouping them into several clusters. Then for each input and

[1] This will be problematic for neural network models. The reason is as follows. The natural choice for optimizing neural network models is to use the stochastic gradient descent method with shuffled mini-batches. But in highly imbalanced data sets, most mini-batches only contain negative samples (i.e., the samples without pivots). These batches will mislead the learning process of the whole model.

each pivot cluster, we predict if the input contains at least one word from that pivot cluster. Considering that our end task is polarity classification, we choose to group the pivots into two lists: a positive pivot word list and a negative pivot word list.

Specifically, we first tokenize the samples in $\mathcal{D}^s$ and $\mathcal{D}^{\text{t,u}}$ and perform part-of-speech tagging using the NLTK toolkit[2]. Next, we extract only adjectives, adverbs and verbs with a frequency threshold K_s in $\mathcal{D}^s$ and K_t in $\mathcal{D}^{\text{t,u}}$. We also remove negation words such as *not* and stop words using a stop word list to obtain a list of pivot candidates $\mathcal{C}$. Then, for each word in $\mathcal{C}$, we measure its mutual information (MI) with the positive and the negative classes based on $\mathcal{D}^s$ as follows:

$$r(w_i, y) = \log \frac{\tilde{p}(w_i, y)}{\tilde{p}(w_i)\tilde{p}(y)},$$

where w_i denotes the i^{th} word in $\mathcal{V}$, $y \in \{+, -\}$ is a sentiment label, and $\tilde{p}(w_i, y)$ is the empirical probability of observing w_i and y together. 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 pivots for the positive and the negative sentiments. Some manual inspection shows that most of these words are indeed domain-independent sentiment words.

Given these two pivot word lists, we now formally define our first auxiliary task. For each input $\boldsymbol{x}$, we replace all the occurrences of these pivots with a special token *UNK*. Let $g(\cdot)$ be a function that denotes this procedure, that is, $g(\boldsymbol{x})$ is the resulting review with *UNK* tokens. We then introduce two binary labels for $g(\boldsymbol{x})$. The first label u indicates whether the original input $\boldsymbol{x}$ contains at least one positive pivot word, and the second label v indicates whether $\boldsymbol{x}$ contains at least one negative pivot word. Figure 4.4 shows an example input $\boldsymbol{x}$, its modified version $g(\boldsymbol{x})$ and the labels u and v for $\boldsymbol{x}$. We further use $\mathcal{D}_1^a = \{(g(\boldsymbol{x}_i), u_i, v_i)\}_{i=1}^{N^a}$ to denote a set of training samples for the auxiliary task, where $\mathcal{D}_1^a$ refers to the samples in $\mathcal{D}^s$ and $\mathcal{D}^{\text{t,u}}$.

[2]http://www.nltk.org/

4.3.2 Auxiliary Task 2

Unlike the first auxiliary task, which still follows SCL to predict the occurrence of pivots, we propose an alternative auxiliary task by predicting whether the sum of all the pivots' sentiment scores is larger than, equal to or less than 0 for each piece of input text[3].

Since this auxiliary task relies on the sentiment scores of pivots, we propose to use MI to automatically derive a pseudo sentiment score for each pivot. More specifically, the same as in Section 4.3.1, we can first extract the pivot candidates $\mathcal{C}$ and rank them in decreasing order based on each word's MI with the positive and the negative classes. Then, we only keep those words with positive MI, i.e., $r(w_i, y) > 0$, and obtain two lists $\mathcal{R}_+$ and $\mathcal{R}_-$. Moreover, for each word $w \in \mathcal{R}_+$, we use its MI score as its sentiment score, while for each word $w \in \mathcal{R}_-$, we reverse its MI score as its sentiment score. Finally, we merge the two word lists to form the pseudo sentiment lexicon, and rescale the sentiment scores into $[-S, S]$. Since we assume that pivots should be sentiment sensitive, we further set a threshold τ to only keep those words with high absolute sentiment scores.

Given the sentiment scores of pivots, let us formally define our second auxiliary task, which only differs from our first auxiliary task in the auxiliary label. Therefore, we introduce an auxiliary label y' for $g(\boldsymbol{x})$, which indicates whether the sum of the sentiment scores of the pivot words in the original input $\boldsymbol{x}$ is larger than, equal to or less than 0. We further use $\mathcal{D}_2^{\text{a}} = \{(g(\boldsymbol{x}_i), y_i')\}_{i=1}^{N^a}$ to denote our modified inputs with the auxiliary labels derived from both $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t,u}}$.

In summary, there are two commonalities shared among our two auxiliary tasks: (1) The label of them can be automatically derived. (2) Since pivots are sentiment sensitive, both of them are closely related to the original sentiment classification task. But they differ from each other in how to model the correlations between non-pivot words and pivot words; the first task uses non-pivots to predict the occurrence

[3]For any sentiment lexicon, we can rescale its original sentiment scores to $[-S, S]$, where S can be any integer, and $-S$ and S respectively refer to the most negative and the most positive scores.

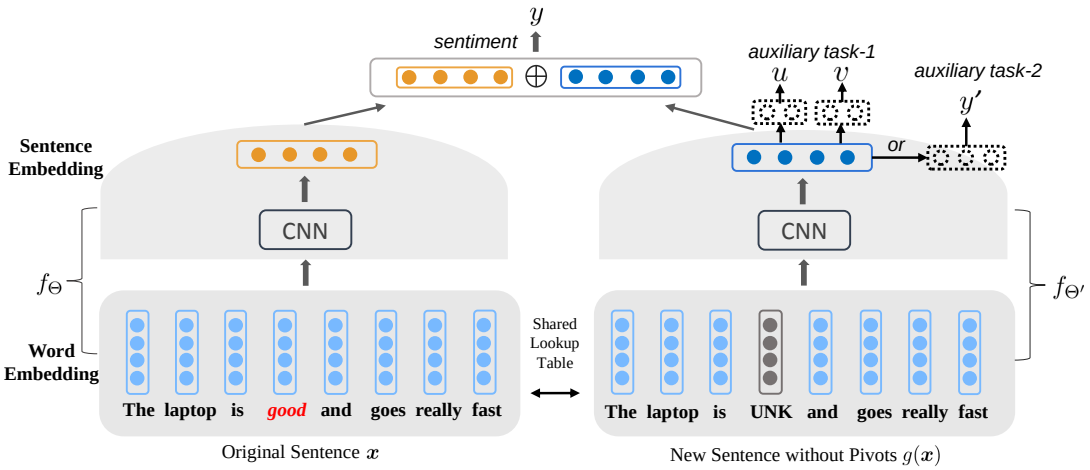

Figure 4.2: Learning Shared Representations for Sentence-level Sentiment Classification.

of pivots, and the second task uses non-pivots to predict the overall sentiment scores of pivots.

4.4 Model

In this section, we present our domain adaptation models for sentence-level and document-level sentiment classification, respectively.

4.4.1 Domain Adaptation for Sentence-Level Sentiment Classification

In this subsection, we focus on sentence-level sentiment classification. We choose to adopt a one-layer CNN [50] to transform an input sentence to a sentence embedding vector because it is simpler than RNN or ReNN and has been shown to work well for sentence-level sentiment classification [50]. Figure 4.4 gives the outline of our model to learn shared sentence embeddings with one of our two auxiliary tasks.

Specifically, each word x_i (including the token *UNK*) in an input $\boldsymbol{x}$ and its modified version $g(\boldsymbol{x})$ is represented by an l-dimensional dense embedding vector, which is retrieved from a lookup table $\mathbf{X} \in \mathbb{R}^{l \times |\mathcal{V}|}$.

With the two auxiliary tasks, we first learn a CNN model to produce sentence

embeddings that work well for the auxiliary tasks. To be consistent with our previous notations, we still use Θ' to denote the parameters of our CNN model that produces the sentence embedding $\mathbf{h}' = \mathit{CNN}_{\Theta'}(g(\boldsymbol{x}))$, where *CNN*[4] stands for the corresponding transformation function f.

Next, if we choose the first auxiliary task, we can employ two softmax classifiers to predict the two auxiliary labels u and v, respectively:

$$\begin{aligned} p(u \mid \mathbf{h}') &= \text{softmax}(\mathbf{W}'_u \mathbf{h}' + \mathbf{b}'_u), \\ p(v \mid \mathbf{h}') &= \text{softmax}(\mathbf{W}'_v \mathbf{h}' + \mathbf{b}'_v), \end{aligned}$$

where $\mathbf{W}'_u \in \mathbb{R}^{2\times m}$ and $\mathbf{W}'_v \in \mathbb{R}^{2\times m}$ are weight matrices, and $\mathbf{b}'_u \in \mathbb{R}^2$ and $\mathbf{b}'_v \in \mathbb{R}^2$ are bias vectors.

Similarly, if we choose the second auxiliary task, a softmax classifier can be learned to map $\mathbf{h}'$ to its auxiliary label y':

$$p(y' \mid \mathbf{h}') = \text{softmax}(\mathbf{W}'\mathbf{h}' + \mathbf{b}'),$$

where $\mathbf{W}' \in \mathbb{R}^{3\times m}$ is a weight matrix, and $\mathbf{b}' \in \mathbb{R}^3$ is a bias vector.

Besides, we also apply another CNN to obtain the standard hidden layer $\mathbf{h} = \mathit{CNN}_{\Theta}(\boldsymbol{x})$.

Finally, we concatenate the standard hidden vector $\mathbf{h}$ and the auxiliary hidden vector $\mathbf{h}'$ to predict the actual sentiment label y:

$$p(y \mid \mathbf{h}, \mathbf{h}') = \text{softmax}\Big(\mathbf{W}(\mathbf{h} \oplus \mathbf{h}') + \mathbf{b}\Big).$$

[4]To simplify the discussion, we will not give the details of CNN here. Interested readers can refer to [50].

4.4.2 Domain Adaptation for Document-Level Sentiment Classification

In this subsection, we focus on document-level sentiment classification, where we first introduce our base model, followed by presenting our proposed two domain adaptation architectures.

Base Model

Instead of using the simple CNN model as before, here we employ a state-of-the-art hierarchical neural network (HNN) model proposed in [103], which first encodes each sentence in the document into a sentence embedding through a CNN, followed by composing all sentence embeddings to a document embedding with a gated RNN (GRNN). The reason for using HNN is that the hierarchical model has been shown to capture semantic relations between sentences and significantly outperform simpler, non-hierarchical models including CNN and RNN in several benchmark data sets [103, 123].

Recall that an input document d is represented by a sequence of sentences, each containing a sequence of words, and $w_{i,j} \in \mathcal{V}$ is the j^{th} word of the i^{th} sentence in d. We use $\mathbf{x}_{i,j} \in \mathbb{R}^l$ to denote an l-dimensional dense embedding vector for word $w_{i,j}$, which is retrieved from a lookup table $\mathbf{X} \in \mathbb{R}^{l \times |\mathcal{V}|}$ for all words. We first apply a one-layer CNN [50] to obtain an embedding vector $\mathbf{z}_i \in \mathbb{R}^p$ for the i^{th} sentence: $\mathbf{z}_i = \mathit{CNN}_{\Theta_1}(\mathbf{x}_i)$, where Θ_1 denotes all the parameters in this CNN.

After obtaining the sentence embeddings for all the n sentences in d, we then apply an LSTM to sequentially combine all sentences together: $\mathbf{h}_i = \mathit{LSTM}_{\Theta_2}(\mathbf{h}_{i-1}, \mathbf{z}_i)$, where $\mathbf{h}_i \in \mathbb{R}^q$ is the i^{th} hidden state, and Θ_2 denotes all the parameters in the LSTM[5]. Note that [103] used bi-directional gated RNN to chain the sentences into a document embedding, but we did not observe any significant gain over LSTM based on our preliminary experiments.

[5]To simplify the discussion, we will not give the details of CNN and LSTM here. Interested Readers can refer to [50] and [41].

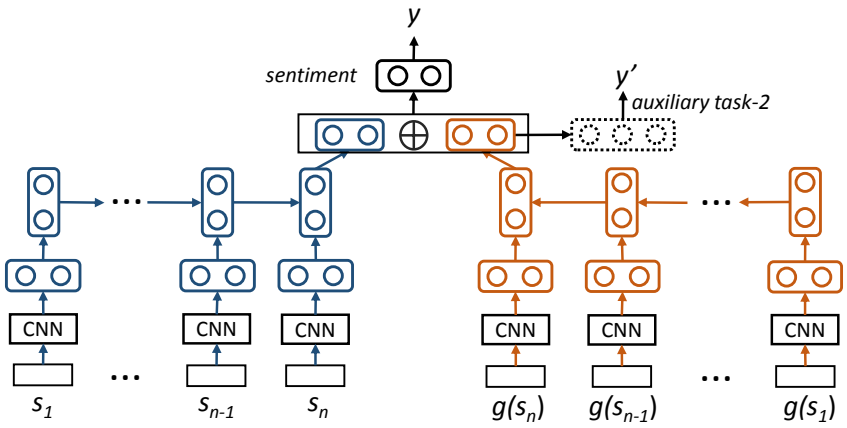

Figure 4.3: Document-Level Shared Representation.

Finally, a softmax classifier is learned to map the document representation $\mathbf{h}_n$ to a label y:

$$p(y \mid \mathbf{h}_n) = \text{softmax}(\mathbf{W}\mathbf{h}_n + \mathbf{b}).$$

where $\mathbf{W} \in \mathbb{R}^{2\times q}$ is a weight matrix and $\mathbf{b} \in \mathbb{R}^2$ is a bias vector.

In the following sections, we will present two NN architectures built on top of HNN that leverage our two auxiliary tasks for domain adaptation of document-level sentiment classification. The first architecture uses document-level auxiliary tasks to help induce a document-level hidden representation, while the second architecture uses sentence-level auxiliary tasks to help induce a sentence-level hidden representation. Before introducing our model, we further assume that an input document $\boldsymbol{x}$ contains n sentences $(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n)$, and its modified version $g(\boldsymbol{x})$ is referred as $\Big(g(\boldsymbol{s}_1), g(\boldsymbol{s}_2), \ldots, g(\boldsymbol{s}_n)\Big)$.

Document-Level Shared Representation (DSR)

Figure 4.3 gives the outline of the first architecture, which essentially tries to directly learn an auxiliary hidden layer for each input document.

For the i^{th} sentence in d', we first use a CNN to obtain its auxiliary embedding vector $\mathbf{z}'_i = \mathit{CNN}_{\Theta'_1}(\mathbf{x}'_i)$. These sentence embeddings are further combined together with an LSTM parameterized by Θ'_2, and the final hidden state $\mathbf{h}'_n$ is fed to softmax

classifiers to either predict the first auxiliary task:

$$\begin{aligned} p(u \mid \mathbf{h}'_n) &= \text{softmax}(\mathbf{W}'_u \mathbf{h}'_n + \mathbf{b}'_u), \\ p(v \mid \mathbf{h}'_n) &= \text{softmax}(\mathbf{W}'_v \mathbf{h}'_n + \mathbf{b}'_v). \end{aligned}$$

or the second auxiliary task:

$$p(y' \mid \mathbf{h}'_n) = \text{softmax}(\mathbf{W}' \mathbf{h}'_n + \mathbf{b}').$$

Besides, we also apply another CNN and LSTM to obtain the standard document representation $\mathbf{h}_n$, and concatenate it with the auxiliary hidden vector $\mathbf{h}'_n$ to predict the sentiment label y:

$$p(y \mid \mathbf{h}_n, \mathbf{h}'_n) = \text{softmax}\Big(\mathbf{W}(\mathbf{h}_n \oplus \mathbf{h}'_n) + \mathbf{b}\Big).$$

Sentence-Level Shared Representation (SSR)

Unlike the first architecture, our second proposal focuses on learning an auxiliary hidden layer for each sentence in a given document. As illustrated in Figure 4.4, instead of using overall auxiliary labels for the whole document, we will have auxiliary labels for each sentence in the document.

To facilitate the discussion, for the i^{th} modified sentence $g(\boldsymbol{s}_i)$, let us introduce two auxiliary labels u'_i and v'_i for the first auxiliary task and an auxiliary label y'_i for the second auxiliary task. We further use $\mathbf{u}' \in \mathbb{R}^n$, $\mathbf{v}' \in \mathbb{R}^n$ and $\mathbf{y}' \in \mathbb{R}^n$ to denote the three auxiliary labels for all the n sentences in $g(\boldsymbol{x})$. Let $\mathcal{D}_1^{\text{a,s}} = \{(g(\boldsymbol{x}_i), \mathbf{u}'_i, \mathbf{v}'_i)\}_{i=1}^{N^a}$ denote input documents with the sentence-level auxiliary labels derived from the first auxiliary task, and $\mathcal{D}_2^{\text{a,s}} = \{(g(\boldsymbol{x}_i), \mathbf{y}'_i)\}_{i=1}^{N^a}$ from the second auxiliary task.

Given the two sentence-level auxiliary tasks, we use two CNNs to obtain sentence embeddings $\mathbf{z}_i = \mathit{CNN}_{\Theta_1}(\boldsymbol{s}_i)$ and $\mathbf{z}'_i = \mathit{CNN}_{\Theta'_1}(g(\boldsymbol{s}_i))$, respectively, for $\boldsymbol{s}_i$ and $g(\boldsymbol{s}_i)$.

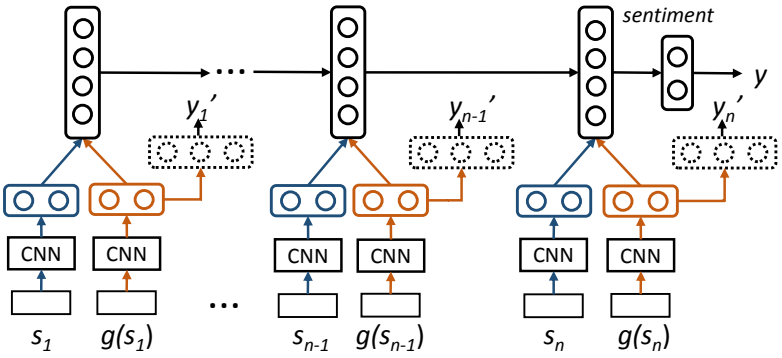

Figure 4.4: Sentence-Level Shared Representation.

Next, the auxiliary hidden layer $\mathbf{z}'_i$ will be used for predicting either the two auxiliary labels u'_i and v'_i:

$$\begin{aligned} p(u'_i \mid \mathbf{z}'_i) &= \text{softmax}(\mathbf{W}'_u \mathbf{z}'_i + \mathbf{b}'), \\ p(v'_i \mid \mathbf{z}'_i) &= \text{softmax}(\mathbf{W}'_v \mathbf{z}'_i + \mathbf{b}'). \end{aligned}$$

or the auxiliary label y'_i:

$$p(y'_i \mid \mathbf{z}'_i) = \text{softmax}(\mathbf{W}' \mathbf{z}'_i + \mathbf{b}').$$

Besides, we also concatenate $\mathbf{z}_i$ and $\mathbf{z}'_i$ together as a combined sentence embedding for the i-th sentence. Then, all the n combined sentence embeddings are further combined together via an LSTM:

$$\mathbf{h}_i = \mathit{LSTM}_{\Theta_2}\Big(\mathbf{h}_{i-1}, (\mathbf{z}_i \oplus \mathbf{z}'_i)\Big).$$

Finally, we feed the last hidden representation $\mathbf{h}_n$ to a softmax classifier to predict the label of our main task:

$$p(y \mid \mathbf{h}_n) = \text{softmax}(\mathbf{W}\mathbf{h}_n + \mathbf{b}).$$

Comparing our two models with the standard HNN model in [103], we can see that there are two hidden layers in the standard model, namely, the lower-level sentence embeddings and the higher-level document embeddings, but they are only

trained on $\mathcal{D}^{\text{s}}$; in our two models, we leverage both $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t,u}}$ to train the auxiliary tasks to respectively induce shared cross-domain document embeddings or sentence embeddings, and concatenate them together with the original document embeddings or sentence embeddings for predicting our main task. Hence, the hidden layers in the standard model are only sensitive to the source domain, while the hidden layers in our models are robust in both domains.

4.4.3 Model Optimization

Since our model consists of the actual sentiment classification task and our proposed auxiliary tasks, we propose to jointly optimize them in a single loss function. For simplicity, here we just use the second auxiliary task to show our objective function, and the objective function of using the first auxiliary task is very similar. In this way, the learning of the parameters corresponding to our auxiliary task 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, using cross-entropy loss, we can learn Θ, Θ', $\mathbf{W}$, $\mathbf{b}$, $\mathbf{W}'$ and $\mathbf{b}'$ in Section 4.4.1 by minimizing the following function through backpropagation:

$$\begin{aligned} & J(\Theta, \Theta', \mathbf{W}, \mathbf{b}, \mathbf{W}', \mathbf{b}') \\ = & -\Big(\sum_{(\boldsymbol{x},y)\in\mathcal{D}^{\text{s}}} \log p(y \mid \mathbf{h}, \mathbf{h}') + \sum_{(g(\boldsymbol{x}),y')\in\mathcal{D}^{\text{a}}} \log p(y' \mid \mathbf{h}') \Big). \end{aligned} \tag{4.1}$$

Similar to Eqn. (4.1), we can minimize the following loss functions to learn Θ_1, Θ'_1, Θ_2, Θ'_2, $\mathbf{W}$, $\mathbf{W}'$, $\mathbf{b}$ and $\mathbf{b}'$ for the first architecture in Section 4.4.2:

$$\begin{aligned} & J(\Theta_1, \Theta'_1, \Theta_2, \Theta'_2, \mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}') \\ = & -\Big(\sum_{(\boldsymbol{x},y)\in\mathcal{D}^{\text{s}}} \log p(y \mid \mathbf{h}_n, \mathbf{h}'_n) + \sum_{(g(\boldsymbol{x}),y')\in\mathcal{D}^{\text{a}}} \log p(y' \mid \mathbf{h}'_n) \Big). \end{aligned}$$

and learn Θ_1, Θ_1', Θ_2, $\mathbf{W}$, $\mathbf{W}'$, $\mathbf{b}$ and $\mathbf{b}'$ for the second architecture in Section 4.4.2:

$$\begin{aligned} & J(\Theta_1, \Theta_1', \Theta_2, \mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}') \\ = & -\Big(\sum_{(\boldsymbol{x},y)\in\mathcal{D}^{\mathrm{s}}} \log p(y \mid \mathbf{h}_n) + \sum_{(g(\boldsymbol{x}),\mathbf{y}')\in\mathcal{D}^{\mathrm{a,s}}} \sum_{i=1}^{n} \log p(y_i' \mid \mathbf{z}_i') \Big). \end{aligned}$$

4.4.4 Differences from SCL

Although our domain adaptation methods are inspired by SCL, there are a number of major differences: (1) Our methods are 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 either two binary classification tasks or one multi-class classification task, while SCL uses much more pivot prediction tasks. (3) We leverage neural network models to directly produce the hidden representation, while SCL relies on SVD to learn the projection function. (4) Our methods perform joint learning of the auxiliary tasks and the end task, i.e., sentiment classification, while SCL performs the learning in a sequential manner.

4.5 Experiments

4.5.1 Data Sets and Experiment Settings

Data Sets: To evaluate our proposed methods, we conduct experiments on both sentence-level and document-level sentiment classification.

For sentence-level sentiment classification, we use five benchmark data sets about product reviews from four domains. *Movie1*[6] and *Movie2*[7] are movie reviews collected by [80] and [95], respectively. *Camera*[8] is a set of reviews of digital prod-

[6] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[7] http://nlp.stanford.edu/sentiment/
[8] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

Task	Data Set	# UnLab	# Lab	# Pos	# Neg	Avg Length	Words
Sentence-level	Movie1 (*MV1*)	7662	3000	1500	1500	19	18765
	Movie2 (*MV2*)	6613	3000	1500	1500	20	16186
	Camera (*CR*)	2770	1000	600	400	19	5340
	Laptop (*LT*)	1549	1000	500	500	16	3610
	Restaurant (*RT*)	2075	1000	600	400	15	4541
Document-level	Book (**B**)	6000	2000	1000	1000	180	41109
	DVD (**D**)	6000	2000	1000	1000	188	44072
	Electronics (**E**)	6000	2000	1000	1000	118	22444
	Kitchen (**K**)	6000	2000	1000	1000	100	18472

Table 4.1: Statistics of our Data Sets.

ucts such as MP3 players and cameras labeled by [43]. *Laptop* and *Restaurant*[9] are laptop and restaurant reviews taken from SemEval 2014 Task 4 and SemEval 2015 Task 12. Since we do not have any unlabeled reviews for all domains and the number of labeled reviews in each domain is different, to better control the experiment setting, we choose a subset of the original labeled reviews as training data. Specifically, from each of *Movie1* and *Movie2*, we choose 1500 positive and 1500 negative reviews as labeled reviews. From each of the other three domains, we choose around 500 positive and 500 negative reviews as labeled reviews[10]. For all the remaining reviews in each domain, we ignore the labels and treat them as unlabeled reviews.

For document-level sentiment classification, we employ a benchmark data set released by [10]. This data set consists of Amazon product reviews from four different domains: *Book*, *DVD*, *Electronics* and *Kitchen*. Each domain has 1000 positive and 1000 negative reviews as well as 17547 unlabeled reviews on average. Since the number of unlabeled reviews in each domain is different, we choose 6000 unlabeled reviews for each domain.

The statistics of each data set are summarized in Table 4.1.

Experiment Settings: For sentence-level sentiment classification, we consider 18 pairs of data sets where the two data sets come from different domains.[11] For

[9]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.

[10]Since the data distributions of *Camera* and *Restaurant* are imbalanced, we choose 600 positive and 400 negative samples for them, which are similar to their original data distributions.

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

neural network-based methods, we choose 100 positive and 100 negative reviews 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.

Following previous studies [76, 138], we consider 12 pairs of source-target domain pairs for document-level sentiment classification. For each pair, all the 2000 labeled reviews from the source domain are treated as training data. We choose 200 positive and 200 negative reviews from the target domain as development data, and the remainder (i.e., 800 + 800 reviews) from the target domain as test data. Note that this is different from our experiments in Chapter 3, since we use all the reviews from the target domain as test data.

For domain adaptation methods, we also use the unlabeled reviews from the target domain during the training stage[12].

4.5.2 Baselines and Hyperparameters

We consider the following baselines based on bag-of-words (BoW) representations for both sentence-level and document-level sentiment classification:

- **Naive** is a non-domain-adaptive baseline.
- **SCL** is our implementation of the Structural Correspondence Learning method, which uses all the non-pivot features to predict the occurrence of each pivot feature and employs SVD on the learned weight vectors to obtain a dense cross-domain representation.
- **mDA** is our implementation of one of the state-of-the-art methods, marginalized denoising auto-encoders [21], which learns a shared hidden representation by reconstructing pivot features.

Since we employ different neural network models for sentence-level and

[12] In our previous conference version [129], we simply treat the test data from the target domain as the unlabeled data, but in this dissertation, we split the data from the target domain into unlabel/test data.

document-level sentiment classification, we will now introduce the neural network-based systems for comparison. First, for sentence-level sentiment classification:

- **CNN** is a non-domain-adaptive baseline based on CNN.
- **C-Aux** is a simple combination of our first auxiliary tasks with **CNN**, which treats the derived two labels of the first auxiliary task as two features and then appends them to the hidden representation learned from **CNN**, followed by a softmax classifier.
- **C-SCL** is a naive combination of **SCL** with **CNN**, which appends the induced representation from **SCL** to the hidden representation learned from **CNN**, followed by a softmax classifier.
- **C-mDA** is similar to **SCL-NN** but uses the hidden representation derived from **mDA**, which is a strong baseline proposed by us.
- **Joint1** and **Joint2** are respectively our proposed domain adaptation methods with the first and the second auxiliary tasks, as introduced in Section 4.4.1.
- **WJoint2** is a slight modification of **Joint2**. Since the sentiment scores of pivots in our second auxiliary task are derived from $\mathcal{D}^{\text{s}}$, they might be inaccurate and specific to the source domain. Hence, in **WJoint2**, we use an external sentiment lexicon called SentiWordNet [5] to derive the label of our second auxiliary task.

Next, for document-level sentiment classification,

- **HNN** is a non-domain-adaptive baseline based on the hierarchical NN proposed by [103].
- **H-WN** is a simple combination of **HNN** with our second auxiliary task, which represents each label derived from our second auxiliary task using SentiWordNet as a three-dimensional one-hot vector and appends it to the document embedding $\mathbf{h}_n$ in **HNN**, followed by a softmax classifier.

- **H-SCL** and **H-mDA** are naive combinations of **SCL** and **mDA** with **HNN**, which are similar to **C-SCL** and **C-mDA**.

- **DSR1** and **DSR2** are our proposed methods of learning document-level shared representations with the two auxiliary tasks respectively, as introduced in Section 4.4.2.

- **SSR1** and **SSR2** are our proposed methods of inducing shared representations at sentence level with the two auxiliary tasks, respectively, as introduced in Section 4.4.2.

- **WDSR2** and **WSSR2** are modified versions of **DSR2** and **SSR2**, where we use SentiWordNet to derive the label of the second auxiliary task.

For **Naive**, we train linear classifiers with LibLinear[13] by using unigrams and bigrams with a frequency of at least 5 as features and use its default hyperparameters. For **SCL** and **mDA**, we follow [10] and use mutual information to select pivot features, and the number of top ranked pivots we choose is tuned from $\{500,1000,1500,2000\}$ on the development set. In **SCL**, we tune the number of induced features K in $\{25,50,100\}$, and also use normalization and rescaling. In **mDA**, we employ the dropout noise strategy in [120] without any parameter.

For neural network-based models, we set the dimension size of word embeddings l to 300 and initialize the lookup table $\mathbf{X}$ with word embeddings pre-trained from *word2vec*[14]. All the word embeddings are updated during our learning process. Also, for our methods, we share the word embeddings of the actual task and our auxiliary task, and never update the word embedding of the special token *UNK* by setting it as a zero vector. For **CNN**, we set the window size to 3, and set the non-linear activation function in CNN as ReLU. Also, the size of the hidden representations $\mathbf{h}$ is set to 100. Moreover, for **HNN**, we also set the window size to 3 and the non-linear activation function in CNN as ReLU, and set the sizes of sentence

[13]http://www.csie.ntu.edu.tw/cjlin/liblinear/
[14]https://code.google.com/p/word2vec/

embeddings $\mathbf{z}$ and document embeddings $\mathbf{h}_n$ as 150. In our proposed methods, we keep the sizes of $\mathbf{h}$, $\mathbf{z}$ and $\mathbf{h}_n$ unchanged; but for $\mathbf{h}'$, $\mathbf{z}'$ and $\mathbf{h}'_n$, we tuned their sizes in the range of [100, 300], and found that using different hidden sizes did not change results significantly. Hence, we set the size of $\mathbf{h}'$ as 100, and set the sizes of $\mathbf{z}'$ and $\mathbf{h}'_n$ as 150. For all the **CNN**-based models, 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. All the **HNN**-based models are trained using AdaGrad with a learning rate of 0.05 and a minibatch size of 5. Also, the dropout rate α equals 0.5, and all the model parameters are regularized with a L2 regularization strength of 10^{-4}.

Besides, for our two auxiliary tasks, we set the word frequency thresholds K_s and K_t as 3 for sentence-level sentiment classification, and set K_s as 10 and K_t as 30 for document-level sentiment classification. Also, for the second auxiliary task, we set the scale of sentiment lexicon as 2, i.e., $S = 2$, and the threshold τ for choosing words with high sentiment scores as 0.9.

4.5.3 Results

In Table 4.2, Table 4.3 and Table 4.4, we report the results of all the methods for sentence-level and document-level sentiment classification, respectively.

First, it is easy to see from Table 4.2 that the performance of **Naive** is very limited. **SCL** and **mDA** can further improve the average accuracy respectively by 2.3 and 5 percentage points, which verifies the usefulness of these two domain adaptation methods. Similar trends can be observed in Table 4.3, where **SCL** and **mDA** can also outperform **Naive** by 2.7% and 3.7% on average. However, we can easily see that the performance of these domain adaptation methods based on discrete, bag-of-words representations is even much lower than the non-domain-adaptive method on continuous representations (**CNN** and **HNN**). But we can further see that the cross-domain performance of **CNN** is still 11.4 percentage points lower than its in-domain performance on average, and similarly, the average performance of **HNN**

Task		Compared Methods							Our Methods		
src	tgt	Naive	SCL	mDA	CNN	C-Aux	C-SCL	C-mDA	Joint1	Joint2	WJoint2
MV1	*LT*	0.638	0.633	0.633	0.750	0.747	0.744	0.756	0.773†	0.767†	**0.775**†
MV2	*LT*	0.679	0.683	0.690	0.772	0.781	0.774	0.798	**0.808**†	0.801	0.806†
RT	*LT*	0.668	0.705	0.740	0.788	0.797	0.796	0.805	**0.809**†	0.807	0.803
CR	*LT*	0.713	0.735	0.768	0.813	0.818	0.820	0.815	0.825†	**0.834**†	0.831†
MV1	*RT*	0.634	0.646	0.660	0.741	0.758	0.750	**0.779**	0.760	0.774	**0.779**
MV2	*RT*	0.679	0.700	0.696	0.793	0.797	0.793	0.799	0.808†	**0.817**†	0.809†
LT	*RT*	0.661	0.700	0.744	0.778	0.781	0.806	0.805	0.801	0.787	**0.818**†
CR	*RT*	0.694	0.719	0.741	0.795	0.802	0.801	0.813	0.813	0.805	**0.814**
MV1	*CR*	0.568	0.588	0.549	0.700	0.708	0.723	0.708	0.734†	0.731†	**0.756**†
MV2	*CR*	0.666	0.663	0.671	0.736	0.740	0.748	0.758	0.765†	**0.767**†	0.755
LT	*CR*	0.689	0.704	0.711	0.805	0.797	0.793	0.780	0.791	**0.817**†	0.804
RT	*CR*	0.674	0.683	0.700	**0.798**	0.771	0.766	0.773	0.780	0.775	0.791
LT	*MV1*	0.575	0.585	0.614	0.680	0.676	0.692	0.695	**0.698**	0.692	0.695
RT	*MV1*	0.591	0.611	0.642	0.712	0.707	0.705	0.711	**0.713**	0.711	0.710
CR	*MV1*	0.567	0.581	0.597	0.688	0.692	0.693	0.691	0.698†	**0.699**†	**0.699**†
LT	*MV2*	0.594	0.612	0.635	0.723	0.738	0.727	0.750	**0.756**†	0.747	0.736
RT	*MV2*	0.615	0.629	0.661	0.767	0.767	0.752	**0.770**	**0.770**	0.763	0.762
CR	*MV2*	0.605	0.613	0.643	0.733	0.748	0.744	0.752	**0.753**	0.744	0.746
Average		0.640	0.655	0.672	0.754	0.757	0.757	0.764	0.770	0.769	**0.772**

Table 4.2: Classification accuracies of our proposed methods with the two auxiliary tasks on sentence-level sentiment classification. † indicates that our joint methods are significantly better than **CNN**, **C-Aux**, **C-SCL** and **C-mDA** with $p < 0.05$ based on McNemar's paired significance test. Note that the in-domain sentiment classification performance of *LT*, *RT*, *CR*, *MV1* and *MV2* are respectively 0.857, 0.847, 0.853, 0.803, 0.840 (0.840 on average) in the setting where we use each target domain's unlabeled reviews as training set, and the development and test sets are the same as our **Domain Adaptation** setting.

in cross-domain setting also drops by 7.8 percentage points compared with its performance in in-domain setting. This indicates that it will be more challenging and useful to develop domain adaptation methods based on embedding vectors and neural network models.

Moreover, we can find that the performance of simply appending BoW-based features from our auxiliary tasks to **CNN** and **HNN** (i.e., **C-Aux** and **H-WN**) is quite close to that of **CNN** and **HNN** on most data set pairs, which shows that this kind of simple combination is not ideal for domain adaptation. In addition, although **SCL** can significantly outperform **Naive** on almost all the data set pairs, the performance of **C-SCL** and **H-SCL** is not satisfactory, which can only improve **CNN** and **HNN** by 0.4 and 0.5 percentage point on average, respectively. But for **C-mDA** and **H-mDA**, although the shared hidden representations are also derived

Task		Compared Methods							Our Methods with Auxiliary Task 1	
src	tgt	Naive	SCL	mDA	HNN	H-WN	H-SCL	H-mDA	DSR1	SSR1
E	**D**	0.680	0.700	0.727	0.805	0.806	**0.820**	0.811	0.798	0.814
B	**D**	0.773	0.771	0.806	0.814	0.832	0.813	0.829	0.823	**0.832**
K	**D**	0.698	0.721	0.741	0.791	0.796	0.799	0.796	0.788	**0.803**†
E	**B**	0.693	0.704	0.728	0.786	**0.790**	0.780	0.789	0.789	0.781
D	**B**	0.751	0.780	0.802	0.805	0.810	0.796	0.818	0.809	**0.826**†
K	**B**	0.690	0.740	0.725	0.766	0.773	0.772	0.774	**0.781**†	0.773
B	**E**	0.701	0.746	0.753	0.755	0.751	0.751	**0.786**	0.758	0.758
D	**E**	0.706	0.743	0.746	0.772	0.768	0.771	**0.786**	0.774	0.782
K	**E**	0.799	0.818	0.830	0.836	0.837	**0.847**	0.839	0.837	0.843
E	**K**	0.828	0.829	0.833	0.852	0.848	**0.865**	0.859	0.864	0.862
B	**K**	0.724	0.763	0.754	0.780	0.788	0.785	**0.798**	0.784	0.791
D	**K**	0.716	0.758	0.742	0.778	0.774	0.786	0.773	0.793†	**0.809**†
Average		0.729	0.756	0.766	0.795	0.798	0.799	0.805	0.800	**0.806**

Table 4.3: Classification accuracies of our proposed methods with the first auxiliary task on document-level sentiment classification. † indicates that our proposed methods are significantly better than **HNN**, **H-WN**, **H-SCL** and **H-mDA**. Note that the in-domain sentiment classification performance of *D*, *B*, *E* and *K* are respectively 0.845, 0.843, 0.858, 0.883 (0.857 on average) in the setting of splitting each domain's labeled reviews into 1400/200/400 as training set, development set and test set.

from traditional BoW representations, they can outperform **CNN** and **HNN** in most cases. This implies that the derived shared representations by **mDA** can generalize better across domains, and are generally useful for domain adaptation.

Furthermore, we can observe the following by comparing our proposed models with the compared systems: (1) For sentence-level sentiment classification, we can see from Table 4.2 that our proposed joint learning methods with the first auxiliary task (i.e., **Joint1**) and the second auxiliary task (i.e., **Joint2** and **WJoint2**) outperform **C-SCL** on almost all the data set pairs. And in comparison with **C-mDA**, **Joint1**, **Joint2** and **WJoint2** can also outperform it on most data set pairs. (2) For document-level sentiment classification, it can be observed from both Table 4.3 and Table 4.4 that all of our proposed methods except **DSR1** can significantly outperform **HNN** in almost all the data set pairs, and perform better than **H-WN** and **H-SCL** in most cases, which shows that the idea of learning a hidden representation using our proposed auxiliary tasks is generally effective. Even compared with **H-mDA**, a strong baseline proposed by us, **SSR1** and **DSR2** can achieve compara-

Task		Compared Methods							Our Methods with Auxiliary Task 2			
src	tgt	Naive	SCL	mDA	HNN	H-WN	H-SCL	H-mDA	DSR2	SSR2	WDSR2	WSSR2
E	**D**	0.680	0.700	0.727	0.805	0.806	0.820	0.811	0.810	**0.823**	0.821	0.816
B	**D**	0.773	0.771	0.806	0.814	0.832	0.813	0.829	0.832	0.822	**0.840**	0.837
K	**D**	0.698	0.721	0.741	0.791	0.796	0.799	0.796	0.798	**0.808**	0.805	0.801
E	**B**	0.693	0.704	0.728	0.786	0.790	0.780	0.789	0.790	0.792	0.790	**0.794**
D	**B**	0.751	0.780	0.802	0.805	0.810	0.796	0.818	**0.835**	0.822	0.825	0.822
K	**B**	0.690	0.740	0.725	0.766	0.773	0.772	0.774	0.774	**0.784**	0.781	0.776
B	**E**	0.701	0.746	0.753	0.755	0.751	0.751	**0.786**	0.760	0.773	0.771	**0.786**
D	**E**	0.706	0.743	0.746	0.772	0.768	0.771	0.786	0.787	0.790	**0.810**	0.799
K	**E**	0.799	0.818	0.830	0.836	0.837	**0.847**	0.839	0.837	0.843	0.830	0.835
E	**K**	0.828	0.829	0.833	0.852	0.848	0.865	0.859	0.859	**0.874**	0.867	0.858
B	**K**	0.724	0.763	0.754	0.780	0.788	0.785	**0.798**	0.785	0.783	0.796	0.794
D	**K**	0.716	0.758	0.742	0.778	0.774	0.786	0.773	**0.809**	0.806	0.800	0.803
Average		0.729	0.756	0.766	0.795	0.798	0.799	0.805	0.806	0.809	**0.813**	0.811

Table 4.4: Classification accuracies of our proposed methods with the second auxiliary task on document-level sentiment classification.

ble results while **SSR2**, **WDSR2** and **WSSR2** can still achieve significantly better performance in most cases. We conjecture that the gains of our proposed methods may come from the sharing between two word embedding lookup tables and joint learning of our auxiliary task and the actual task.

Finally, combining Table 4.3 and Table 4.4, we can have the following observations by comparing our proposed two auxiliary tasks: (1) For sentence-level sentiment classification, the two auxiliary tasks **Joint1** and **Joint2** can achieve similar performance on average, which shows that both of them are useful in this setting. (2) For the **DSR** model in the document-level sentiment classification, using of the first auxiliary task **DSR1** fails to bring improvement in most cases, while the second auxiliary task **DSR2** can outperform most baseline methods. The reason for this is as follows. In document-level sentiment classification, a document may contain mixed opinions towards different aspects of the topic, and the sentiment polarities towards different aspects may differ. Hence, it is highly possible for a document to contain both positive and negative domain-independent sentiment words. In this case, the first auxiliary task would not be of much use because most documents would have the same auxiliary labels[15]. But our second auxiliary task does not have

[15]We found that in our data set, for almost all the 12 source/target pairs, over 90% of the reviews contained both positive and negative pivot words.

such an issue, which indicates that our second auxiliary task is more effective for the **DSR** model. (3) For the **SSR** model in the document-level sentiment classification, although the performance the second auxiliary task **SSR2** slightly better than **SSR1** on average, both of them can work well and generally outperform most baselines methods. (4) For the second auxiliary task, as we can see from Table 4.2 and Table 4.4, with the help of the external sentiment lexicon SentiWordNet, **WJoint2** can further boost the performance of **Joint2**, and **WDSR2** and **WSSR2** can also perform better than **DSR2** and **SSR2** on average. The reason for this observation is intuitive: the **MI**-based sentiment scores, derived from $\mathcal{D}^s$, are inaccurate and specific to the source domain, while the sentiment scores in SentiWordNet are more accurate and general across domains. Moreover, we can also see that the gap between **SSR2** and **WSSR2** is much smaller than the gap between **DSR2** and **WDSR2**. This suggests that our **DSR2** method is more sensitive to the quality of sentiment lexicons, and with a high quality sentiment lexicon, it can perform best on average.

4.5.4 Case Study

To obtain a better understanding of our method, we first use **Joint2** to conduct a case study on sentence-level sentiment classification, where the source is *MV2* and the target is *LT*.

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 4.5, we present the most useful trigrams of each polarity extracted by **CNN** and by the two components of our joint method. **Joint2-original** refers to the CNN corresponding to CNN_{Θ} while **Joint2-auxiliary** refers to the CNN corresponding to $CNN_{\Theta'}$, which is related to the auxiliary tasks.

In Table 4.5, we can easily observe that for **CNN**, the most important trigrams

Method	Negative Sentiment	Positive Sentiment
CNN	problem_*_*, useless_*_*, disappointing_*_*, *_*_worse, has_a_bad, drive_went_bad, slow_*_*, a_nightmare_*, speaker_did_not, **lot_of_trouble**	*_*_great, best_!_*, *_*_good, *_*_love, am_very_good, is_great_, is_perfect_*, am_very_happy, *_*_awesome, is_beautiful_!
Joint2-original	problem_*_*, useless_*_*, disappointing_*_*, **crashed_*_***, went_bad_and, a_nightmare_*, horrible_*_*, cheap_*_*, it_'s_not, **is_irreplaceable_***	is_great_, *_*_good, *_*_best, I_love_the, is_awesome_*, **is_easy_to**, works_beautifully_*, **easy_*_***, of_my_favorite, **a_very_long**,
Joint2-auxiliary	useless_*_*, disappointing_*_*, **crashed_*_***, problem_*_*, **not_work_***, ***_returned_it**, **is_irreplaceable_***, **shut_down_***, **hot_*_***, a_long_time, cheap_*_*	**easy_to_use**, UNK_it_!, I_'ve_had, **no_problem_***, **a_very_long**, **is_solid_***, **is_very_responsive**, **very_fast_and**, **superior_product_!**

Table 4.5: Comparison of the most useful trigrams chosen by **Joint2** and by **CNN** on *MV2→LT*. Here * denotes a "padding", which we added at the beginning and the end of each sentence. The domain-specific sentiment words are in **bold**.

are domain-independent, which contain some general sentiment words like *good*, *great* and *disappointing*. For our joint model, since **Joint2-auxiliary** is jointly learnt with the sentiment classification task, it is easy to see that most of its extracted trigrams are target-specific sentiment expressions like *crashed*, *easy to use* and *a very long*. Also, for **Joint2-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, to explore how our proposed models help to improve the performance of the standard models like **CNN** and **HNN** in the test data set, we also conduct further analysis on *MV2→LT* and *B→D* to get a deeper insight of our joint models **Joint2** and **DSR2**. Specifically, we sample several samples from the test data set, i.e., *Laptop* and *DVD*. As shown in Table 4.6, for *MV2→LT*, we can easily see that although **CNN** correctly predicts the sentiments of the top two test sentences, it gives wrong predictions on another three test sentences containing ***long*** since it tends to express a negative sentiment in the source *Movie* domain. Similarly, for *B→D*, **HNN** gives the correct prediction on the first document but wrong predictions on another two

MV2→LT	**Review**	**CNN**	**Joint2**
Unlabel Data	**Good** *keyboard,* ***long*** *battery life, largest hard drive.*	-	-
	I don't use my laptop in a way though that needs a ***long*** *battery life so it's* **perfect** *for me.*	-	-
	Toshiba is a **great** *brand, even though I haven't had it for a* ***long*** *time, I am very* **happy** *with it!*	-	-
Test Data	*Lightweight,* ***long*** *battery life, excellent transition from PC;*	1	1
	The battery is really ***long****.*	0	1
	It is light and the battery last a very ***long*** *time.*	0	1
	Very ***long*** *life battery (up to 10-11 hours depending on ...)*	0	1

B→D	**Review**	**HNN**	**DSR2**
Unlabel Data	*Very* **good** *film with a* **great** *cast. Reese and wahlberg are* **wonderful** *in ..... Very much* ***worth watching*** */ owning.*	-	-
	A **great** *movie! What an all star cast! This movie is* ***worth watching*** *over and over again.*	-	-
	I found this dvd to be **engaging** *...* **Fantastic***. Loads of deleted scenes that are very* ***worth watching****.*	-	-
Test Data	*Definitely a great movie, rules ..... It is a movie definitely* ***worth watching*** *..... Strongly recommended with .....*	1	1
	If your not already hooked on the story of these ..... Its not your typical medical drama and certainly ***worth watching****.*	0	1
	..... The plot is not very intriguing..... So no wonder you can not compete with him. Anyway, this film is certainly ***worth watching*** *as a family entertainment!*	0	1

Table 4.6: Examples drawn from *MV2→LT* and **B→D** whose sentiment labels are incorrectly predicted by the baseline models (**CNN** and **HNN**) but correctly inferred by our domain adaptation models (**Joint2** and **DSR2**). The sentiment words specific to the target domain are in **bold** and *italic*, and the pivot sentiment words are only in **bold**. 0 and 1 represent the negative and positive sentiments.

documents containing ***worth watching***, since ***worth watching*** only occurs once in the source *BOOK* domain. However, our joint model **Joint2** and **DSR2** make correct predictions for all these test samples. The reason is as follows. In Table 4.6, we can observe that in the unlabeled data from the *Laptop* domain and the *DVD* domain, ***long*** and ***worth watching*** often co-occur with some general positive sentiment words like *good*, *great*, *wonderful* and *fantastic*. Based on these unlabeled

sentences, our joint models **Joint2** and **DSR2** can implicitly learn that ***long*** and ***worth watching*** are highly correlated with the positive sentiment via our auxiliary task, and ultimately make correct predictions for all the test samples. This further indicates that our joint models can identify more domain-specific sentiment words in comparison with the standard model without domain adaptation, and therefore improve the performance.

4.6 Discussion

In this chapter, we presented a general NN-based domain adaptation framework for sentiment classification. Under the framework, we first devised two new auxiliary tasks based on domain-independent sentiment words. Then, we employed two existing neural network architectures to respectively induce shared sentence embeddings for sentence-level sentiment classification and shared document/sentence embeddings for document-level sentiment classification. Experiment results on several benchmark data sets show that most of our proposed models can outperform several highly competitive domain adaptation methods, and with the help of an external sentiment lexicon, our best model can even achieve the state-of-the-art result.

Part II

Supervised Transfer Learning

Chapter 5

A Supervised Neural Domain Adaptation Framework via Modeling Domain Relationships for Retrieval-based Question Answering Systems

In this chapter, we study supervised domain adaptation for the retrieval-based question answering systems, which can be simplified as the well-studied paraphrase identification (PI) or natural language inference (NLI) tasks. We aim to propose a general framework for PI and NLI, which can effectively and efficiently adapt the *shared* knowledge learned from a resource-rich source domain to a resource-poor target domain. Specifically, since most existing supervised domain adaptation methods only focus on learning a shared feature space across domains while ignoring the relationship between the source and target domains, we propose to simultaneously learn shared representations and domain relationships in a unified framework. Furthermore, we propose an efficient and effective hybrid model by combining a sentence encoding-based method and a sentence interaction-based method as our

base model. Extensive experiments on both paraphrase identification and natural language inference demonstrate that our base model is efficient and has promising performance compared to the competing models, and our supervised domain adaptation method can help to significantly boost the performance. Further analysis shows that the *inter-domain* and *intra-domain* relationship captured by our model are insightful. Last but not least, we deploy our full domain adaptation model for PI into our online chatbot system, which can bring in significant improvements over our existing system. Finally, we launch our new system on the chatbot platform Eva[1] in our E-commerce site *AliExpress*[2].

5.1 Introduction

Question Answering (QA) systems have been widely developed and used in many domains. Examples of industry applications include Alibaba's AliME [87, 59], Microsoft's SuperAgent [25], Apple's Siri and Google's Google Assistant. Generally speaking, there are two kinds of commonly-used techniques behind most QA systems: Information Retrieval (IR)-based models [118] and generation-based models [105]. In this work, we focus on building up an IR-based QA system for automatically answering frequently asked questions (FAQs) in the E-commerce industry.

Figure 5.1 illustrates the workflow of IR-based chatbot systems, where a key component is the *Question Rerank* module which reranks candidate questions in a question-answering knowledge base (KB) to find the best matching question given a question from a user. This task can be reduced to a paraphrase identification or a natural language inference problem. Take the query question and knowledge base shown in Figure 5.1 for example. If we can detect that question C1 in the KB is a paraphrase of the query question, then we can take its answer as the answer for the query. In some cases, if we allow the matching question to be more general than the

[1] `https://gcx.aliexpress.com/ae/evaenglish/portal.htm?pageId=195440`

[2] `http://www.aliexpress.com/`

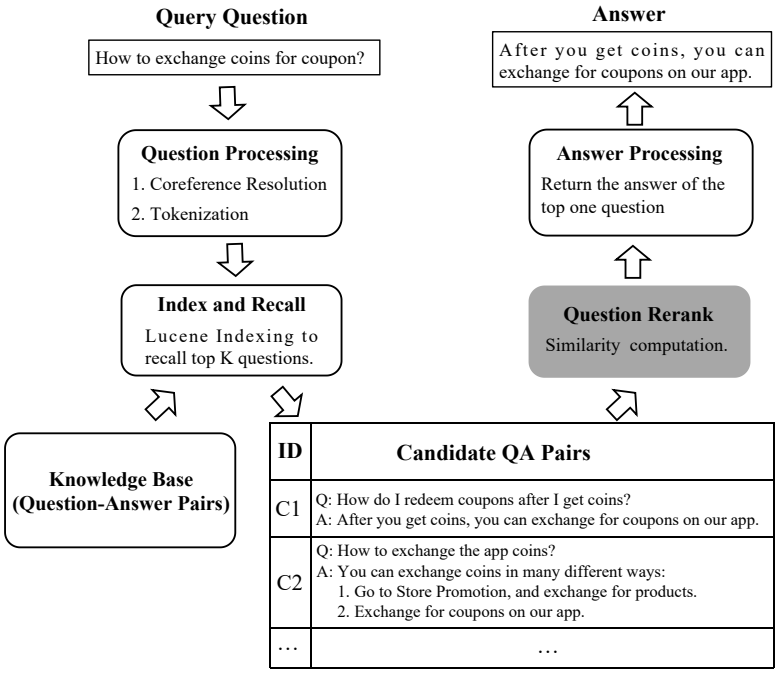

Figure 5.1: The Workflow of IR-based Chatbot Systems.

query question (i.e., entailed by the query question), we can also take the answer for question C2 in the KB as the query question's answer.

In the literature, paraphrase identification (PI) and natural language inference (NLI) have been extensively studied in the last decade [94, 17, 125, 18, 22]. However, when applying existing solutions to PI and NLI in chatbot systems in the E-commerce industry, there are at least two major challenges we face: (1) *Lack of rich training data*: All these solutions rely on a large amount of labeled data. However, it is generally time-consuming and costly to manually annotate sufficient labeled data for each domain. For example, different product categories might need different training data. (2) *Hard to reach a high QPS*[3]. Most of the existing methods focus on improving the effectiveness or accuracy without paying much attention to efficiency. For real industry applications, when real-time responses are expected and a large number of customers are being served simultaneously, we need an efficient method to support a high QPS.

In this chapter, we try to address the two challenges above. Specifically, we first make an empirical comparison of both the effectiveness and efficiency of several

[3]Queries Per Second

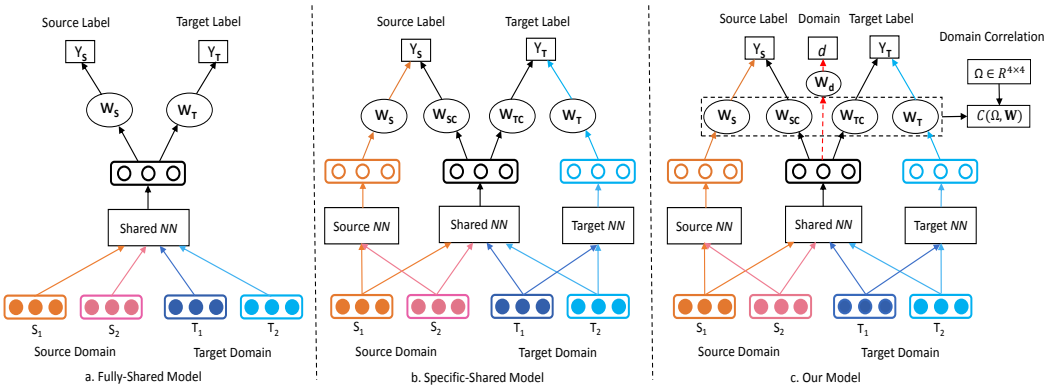

Figure 5.2: Existing Supervised Domain Adaptation Frameworks.

representative methods for modeling sentence pairs and propose an effective and efficient *hybrid model* as our base model. This ensures that we can achieve a high QPS. On top of the base model, we further design a new *Supervised Domain Adaptation (SDA) framework*, which is able to efficiently improve the performance on a resource-poor target domain by leveraging knowledge from a resource-rich source domain.

A Hybrid Base Model. Observing that LSTM-based methods [17, 22] are much more time-consuming than CNN-based methods [125, 69], we focus on CNN-based methods in this study. Meanwhile, there are typically two types of CNN-based methods for the task, namely sentence encoding (SE)-based methods [124, 70] and sentence interaction (SI)-based methods [42, 82]. We argue that these two types of methods may highly complement each other, and thus we propose a hybrid CNN model by combining an SE-based method [125] and an SI-based method [82]. Specifically, we modify the SE-based method using two element-wise comparison functions inspired by [69, 108] to match the two sentence embeddings, and then concatenate them together with sentence embeddings from the SI-based method.

Domain Adaptation Framework. Domain adaptation (DA) aims to apply knowledge gained in a source domain to help a target domain [78]. The key issue is how to transfer the shared knowledge from the source domain to the target domain while exclude the specific knowledge in the source domain based on the *domain relationship*. Most recent studies for DA in NLP perform *multi-task feature*

learning by exploiting different NN models to capture a shared feature space across domains. As illustrated in Figure 5.2a and Figure 5.2b, one line of work employs a fully-shared framework to learn a shared representation followed by using two different fully connected layers for each domain [70, 122], while another line of work uses a specific-shared framework to learn not only a shared representation for both domains but also a domain-specific representation for each domain [64].

However, the first line of work simply assumes that two domains share the same feature space but ignore the domain-specific feature space. Although the latter one is capable of capturing both the shared and the domain-specific representations, it does not consider any relationships between the weights of the final output layer. Generally speaking, the weights on the output layer should capture both the *inter-domain* and the *intra-domain* relationships: (1) For the shared feature space across domains, since it is expected to be domain-independent, the weights corresponding to this feature space in the two domains should be positively related to each other; (2) For the shared and the domain-specific feature spaces in each domain, since they are expected to respectively capture domain-independent and domain-dependent features, their corresponding weights should be irrelevant to each other. Motivated by such an intuition, in this chapter, we propose a new domain adaptation method by explicitly modeling the domain relationships via a covariance matrix, which imposes a regularization term on the weights of the output layer to uncover both the inter-domain and the intra-domain relationships. Besides, to make the shared representation more invariant across domains, we follow some recent work on adversarial networks [36, 64] and introduce an adversarial loss on the shared feature space in our method. Figure 5.2c gives an outline of our full model.

To evaluate our proposed method, we conduct both intrinsic evaluation and extrinsic evaluation.

Intrinsic Evaluation. We conduct extensive experiments on both a benchmark dataset and our own dataset. (1) The hybrid CNN model is shown to be not only efficient but also effective, in comparison with several representative methods; (2) Our

proposed domain adaptation method can bring significant improvements over the base model without domain adaptation, and outperform existing SDA frameworks including the widely used fully-shared model and the recently proposed specific-shared model; (3) Further analysis on our learned correlation matrix shows that our method is able to capture the inter-domain and intra-domain relationships.

Extrinsic Evaluation. We deploy our proposed hybrid CNN-based domain adaptation model into our online chatbot system, which is deployed on a real E-commerce site *AliExpress*. Both the offline and the online evaluations show that our new system can significantly outperform the existing online chatbot system. Finally, we launch our new system on Eva[4], a chatbot platform in *AliExpress*.

5.2 Model

In this section, we present our general model for paraphrase identification and natural language inference, which will be used for question reranking in our chatbot-based QA system.

5.2.1 Problem Formulation and Notation

Our model is designed to address the following general problem. Given a pair of sentences, we would like to identify their semantic relation. For paraphrase identification (PI), the semantic relation indicates whether or not the two sentences express the same meaning [124]; for natural language inference (NLI), it indicates whether a hypothesis sentence can be inferred from a premise sentence [17].

Formally, assume there are two sentences $\mathbf{X}_1 = (\mathbf{x}_1^1, \mathbf{x}_2^1, \ldots, \mathbf{x}_m^1)$ and $\mathbf{X}_2 = (\mathbf{x}_1^2, \mathbf{x}_2^2, \ldots, \mathbf{x}_n^2)$, where $\mathbf{x}_i^j$ denotes an l-dimensional dense embedding vector retrieved from a lookup table $\mathbf{E} \in \mathbb{R}^{l \times |\mathcal{V}|}$ for all the words in the vocabulary $\mathcal{V}$. Our task is to predict the semantic label y which indicates the relation between $\mathbf{X}_1$ and

[4]Eva can be accessed via the following link: `https://gcx.aliexpress.com/ae/evaenglish/portal.htm?pageId=195440`

$\mathbf{X}_2$. For PI, we assume the label y to be either *paraphrase* or *not paraphrase*; for NLI, we assume y to be either *neutral*, *entailment* or *contradiction*.

We consider a domain adaptation setting, where we have a set of labeled sentence pairs from a source domain and a target domain, respectively, denoted by $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$. Note that $|\mathcal{D}^{\text{s}}|$ is assumed to be much larger than $|\mathcal{D}^{\text{t}}|$. We seek to use both $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ to train a good model so that it can work well in the target domain.

To solve such a problem, a widely used supervised domain adaptation (SDA) method (as illustrated in Figure 5.2a) is to use the same NN model to transform every pair of input sentences in both domains into a hidden representation $\mathbf{z_c} \in \mathbb{R}^q$, where q is the size of the hidden representations. To facilitate our discussion, let us assume $\mathbf{z_c} = f_{\Theta_c}(\mathbf{X}_1, \mathbf{X}_2)$, where f_{Θ_c} denotes the transformation function parameterized by Θ_c. Next, for the source and the target domains, we assume that two fully connected layers are separately learned to map $\mathbf{z_c}$ to label y.

$$p(y \mid \mathbf{z_c}) = \begin{cases} \text{softmax}(\mathbf{W_{sc}}\mathbf{z_c} + \mathbf{b_s}) & \text{if } y \text{ is source label,} \\ \text{softmax}(\mathbf{W_{tc}}\mathbf{z_c} + \mathbf{b_t}) & \text{if } y \text{ is target label,} \end{cases}$$

where $\mathbf{W_{sc}} \in \mathbb{R}^{|Y| \times q}$ and $\mathbf{W_{tc}} \in \mathbb{R}^{|Y| \times q}$ are weight matrices and $\mathbf{b_s} \in \mathbb{R}^{|Y|}$ and $\mathbf{b_t} \in \mathbb{R}^{|Y|}$ are bias vectors.

Besides, another SDA approach was recently proposed to use a domain-shared NN model and two domain-specific NN models to obtain a shared embedding $\mathbf{z_c}$ and two domain-specific embeddings $\mathbf{z_s} = f_{\Theta_s}(\mathbf{X}_1, \mathbf{X}_2)$ and $\mathbf{z_t} = f_{\Theta_t}(\mathbf{X}_1, \mathbf{X}_2)$. Therefore, the output layers are defined as:

$$p(y \mid \mathbf{z_c}, \mathbf{z_s}) = \begin{cases} \text{softmax}(\mathbf{W_{sc}}\mathbf{z_c} + \mathbf{W_s}\mathbf{z_s} + \mathbf{b_s}) & \text{if } y \text{ is source label,} \\ \text{softmax}(\mathbf{W_{tc}}\mathbf{z_c} + \mathbf{W_t}\mathbf{z_t} + \mathbf{b_t}) & \text{if } y \text{ is target label.} \end{cases}$$

The main limitation of the fully-shared framework is that it ignores source-specific or target-specific features. While for the specific-shared framework, it

fails to consider any inherent correlations between the weights on the output layers. Therefore, we will introduce our proposed method that explicitly incorporates such correlations into the specific-shared framework in the next session.

5.2.2 Proposed Domain Adaptation Method

Our goal is to model the inter-domain relationship between $\mathbf{W_{sc}}$ and $\mathbf{W_{tc}}$, and the intra-domain relationship between $\mathbf{W_s}$ and $\mathbf{W_{sc}}$ as well as $\mathbf{W_t}$ and $\mathbf{W_{tc}}$. Hence, we first reshape each weight matrix into a vector $\mathbf{w_d} \in \mathbb{R}^{q|Y|}$, followed by concatenating all the four reshaped vectors to form a new matrix $\mathbf{W} \in \mathbb{R}^{q|Y|\times 4}$, where each column corresponds to one weight matrix of the output layer.

Next, to capture the domain relationships mentioned above, we introduce a covariance matrix $\mathbf{\Omega} \in \mathbb{R}^{4\times 4}$. Note that each element $\mathbf{\Omega_{i,j}}$ indicates the correlation between $\mathbf{W_i}$ and $\mathbf{W_j}$, where $\mathbf{i}$ and $\mathbf{j}$ are one of $\mathbf{s}$, $\mathbf{sc}$, $\mathbf{t}$ and $\mathbf{tc}$. Inspired by a general multi-task relationship learning framework as introduced in [135], we consider confining the output layer's weights with $\mathbf{\Omega}$ by using $\mathbf{tr}(\mathbf{W\Omega^{-1}W}^T)$, where $\mathbf{tr}(\cdot)$ is the trace of a square matrix. This means that if $\mathbf{\Omega_{i,j}}$ is a large positive/negative value, $\mathbf{W_i}$ and $\mathbf{W_j}$ will be positively/negatively related to each other; otherwise if $\mathbf{\Omega_{i,j}}$ is close to zero, $\mathbf{W_i}$ and $\mathbf{W_j}$ will be irrelevant to each other. Note that to the best of our knowledge, we are the first to apply the multi-task relationship learning framework into NN-based domain adaptation methods.

In order to simultaneously learn our model parameters and the domain relationships in a unified framework, we formulate our loss function as follows:

$$\begin{aligned}\mathcal{L} = \sum_{\mathbf{k}\in\mathbf{s},\mathbf{t}} -\frac{1}{n_{\mathbf{k}}}\sum_{i=1}^{n_{\mathbf{k}}} \log p\big(y_i \mid \mathbf{X}_1^i, \mathbf{X}_2^i\big) + \frac{\lambda_1}{2}\mathbf{tr}(\mathbf{W\Omega^{-1}W}^T) \\ + \frac{\lambda_2}{2}||\mathbf{W}||_F^2 + \frac{\lambda_3}{2}||\Theta_c||_F^2 + \frac{\lambda_4}{2}||\Theta_s||_F^2 + \frac{\lambda_5}{2}||\Theta_t||_F^2 \\ \text{s.t.} \quad \mathbf{\Omega} \geq 0, \quad \mathbf{tr}(\mathbf{\Omega}) = 1. \end{aligned} \tag{5.1}$$

where λ_1, λ_2, λ_3, λ_4 and λ_5 are regularization parameters, $\mathbf{\Omega}$ is required to be pos-

itive semi-definite, and $\mathbf{tr}(\mathbf{\Omega})$ is required to be 1 without losing generality. In the above formulation, the first term refers to the cross-entropy loss for both domains, and the second term serves as a domain-relationship regularizer to constrain the weights on the output layer. The remaining terms are standard L2-regularization terms.

5.2.3 Adversarial Loss

Our above domain adaptation method is based on the specific-shared framework, which is assumed to well capture the shared and domain-specific feature spaces. However, as suggested by [64], the shared representation learned in this framework may still contain noisy domain-specific features. Therefore, to eliminate the noisy features, here we also consider incorporating an adversarial loss on the shared feature space so that the trained model can not distinguish between the source and target domains on it [36].

First, we assume that the shared layer $\mathbf{z_c}$ is mapped to a binary domain label d, which indicates whether $\mathbf{z_c}$ comes from the source or the target domain:

$$p(d \mid \mathbf{z_c}) = \text{softmax}(\mathbf{W_d z_c} + \mathbf{b_d}).$$

Since the goal of adversarial training is to encourage the shared feature space indiscriminate across two domains, we define the adversarial loss as minimizing the negative entropy of the predicted domain distribution, which is different from maximizing the negative cross-entropy as in [36, 64]:

$$\boldsymbol{\ell} = \sum_{\mathbf{k} \in \mathbf{s}, \mathbf{t}} \Big(\frac{1}{n_{\mathbf{k}}} \sum_{i=1}^{n_{\mathbf{k}}} \sum_{j=0}^{1} p(d_{ij} \mid \mathbf{X}_1^i, \mathbf{X}_2^i) \log p(d_{ij} \mid \mathbf{X}_1^i, \mathbf{X}_2^i) \Big). \tag{5.2}$$

Finally, we obtain a combined objective function as follows:

$$\min_{\substack{\mathbf{\Omega},\mathbf{W},\mathbf{W_d},\Theta_c,\\ \Theta_s,\Theta_t,\mathbf{b_s},\mathbf{b_t},\mathbf{b_d}}} \mathcal{L} + \lambda_0 \ell$$
$$\text{s.t.} \qquad \mathbf{\Omega} \geq 0, \quad \mathbf{tr}(\mathbf{\Omega}) = 1,$$

where λ_0 is a hyper-parameter for tuning the importance of the adversarial loss. As suggested by [135], it is not easy to optimize such a semi-definite programming problem. We will present an alternating training approach in Section 5.2.5 for solving it efficiently.

5.2.4 Base Model

Although the proposed domain adaptation method is general and any neural networks for modeling a pair of sentences can be applied to it, we further target at proposing an efficient and effective base model for encoding a pair of sentences.

On one hand, although various attention-based LSTM architectures have been proposed to achieve a superior performance on both PI and NLI [92, 83, 108, 22], these models are very time-consuming due to the computation of memory cells and attention weights in each time step, which may not satisfy the industry demand, especially when QPS is high. On the other hand, CNN-based models are proven to be efficient, hence are the focus of our study. Most existing CNN-based models can be categorized into two groups: sentence encoding (SE)-based methods and sentence interaction (SI)-based methods. The former aims to first learn good representations for each sentence, followed by using a comparison function to transform them into a single representation [124, 70], while the latter tries to directly model the interaction between two sentences at the beginning and then makes abstractions on top of the interaction output [42, 82]. Observing that the two lines of methods focus on different perspectives to model sentence pairs, we expect that a combination of them can capture both good sentence representations and rich interaction structures.

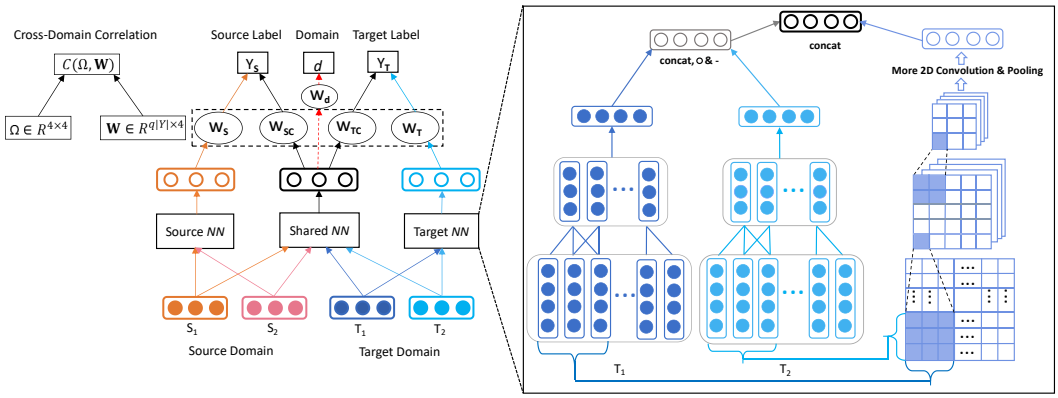

Figure 5.3: Our Full Domain Adaptation Model for Paraphrase Identification and Natural Language Inference.

Hence, we propose a hybrid CNN (hCNN) model, which are based on some minor modifications of two existing models: a SE-based BCNN model [125] and a SI-based Pyramid model [82]. Figure 5.3 depicts our full domain adaptation framework, which contains one shared hCNN and two domain-specific hCNNs. Below we briefly go through the architecture of hCNN. Note that in our implementation and the model description below, we pad the two input sentences to the same length m.

Modified BCNN: Following the original BCNN [125], we first use two separate 1-D convolutional (conv) and 1-D max-pooling layers to encode the two input sentences into two sentence embeddings:

$$\mathbf{h_1} = \mathit{CNN}(\mathbf{X}_1); \quad \mathbf{h_2} = \mathit{CNN}(\mathbf{X}_2).$$

Furthermore, as suggested by [69, 108] that element-wise comparison can work well on the problem, we use two comparison functions to match the two sentence embeddings, and then concatenate them together with the sentence embeddings as the sentence pair representation:

$$\mathbf{H_b} = \mathbf{h_1} \oplus \mathbf{h_2} \oplus (\mathbf{h_1} - \mathbf{h_2}) \oplus (\mathbf{h_1} \cdot \mathbf{h_2}),$$

where $-$ and $\cdot$ refer to element-wise subtraction and element-wise multiplication,

and $\oplus$ refers to concatenation. Note that this setting is different from the original BCNN, which yields better performance in our empirical experiments.

Pyramid: As shown in the rightmost part of Figure 5.3, we first produce an interaction matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$, where $\mathbf{M_{i,j}}$ denotes the similarity score between the i^{th} word in $\mathbf{X}_1$ and the j^{th} word in $\mathbf{X}_2$. Following [82], we use dot-product to compute the similarity score.

Next, by viewing the interaction matrix as an image, we stack two 2-D convolutional layers and two 2-D max-pooling layers on it to obtain the hidden representation $\mathbf{H_p}$.

Finally, we concatenate the two hidden representations as the final representation for each input sentence pair:

$$\mathbf{z} = \mathbf{H_b} \oplus \mathbf{H_p}.$$

5.2.5 Inference

In our combined objective function, we have nine parameters $\mathbf{\Omega}$, $\mathbf{W}$, $\mathbf{W_d}$, $\mathbf{b_s}$, $\mathbf{b_t}$, $\mathbf{b_d}$, Θ_c, Θ_s and Θ_t, and it is not easy to optimize them at the same time. Hence, we employ an alternating stochastic method, i.e., first optimizing the other eight parameters by fixing $\mathbf{\Omega}$, and then alternatively optimizing $\mathbf{\Omega}$ by fixing the others in each iteration. The details are given as below:

Updating $\mathbf{W}$, $\mathbf{W_d}$, $\mathbf{b_s}$, $\mathbf{b_t}$, $\mathbf{b_d}$, Θ_c, Θ_s and Θ_t. While fixing $\mathbf{\Omega}$, the optimization problem becomes:

$$\begin{aligned}
&\min_{\substack{\mathbf{W},\mathbf{W_d},\Theta_c,\Theta_s,\\ \Theta_t,\mathbf{b_s},\mathbf{b_t},\mathbf{b_d}}} \sum_{\mathbf{k}\in\mathbf{s},\mathbf{t}} \Big(\frac{1}{n_\mathbf{k}} \sum_{i=1}^{n_\mathbf{k}} \big(-\log p(y_i \mid \mathbf{X}_1^i, \mathbf{X}_2^i) \\
&+ \lambda_0 \sum_{j=0}^{1} p(d_{ij} \mid \mathbf{X}_1^i, \mathbf{X}_2^i) \log p(d_{ij} \mid \mathbf{X}_1^i, \mathbf{X}_2^i) \big) \Big) + \frac{\lambda_1}{2} \mathbf{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) \\
&+ \frac{\lambda_2}{2}||\mathbf{W}||_F^2 + \frac{\lambda_3}{2}||\Theta_c||_F^2 + \frac{\lambda_4}{2}||\Theta_s||_F^2 + \frac{\lambda_5}{2}||\Theta_t||_F^2
\end{aligned}$$

Algorithm 1 Training Procedure for our Full Model

Input: source training data $\mathcal{D}^{\mathrm{s}}$, target training data $\mathcal{D}^{\mathrm{t}}$ ($|\mathcal{D}^{\mathrm{s}}| \gg |\mathcal{D}^{\mathrm{t}}|$).
Output: $\mathbf{\Omega}$, $\mathbf{W}$, $\mathbf{W_d}$, $\mathbf{b_s}$, $\mathbf{b_t}$, $\mathbf{b_d}$, Θ_c, Θ_s, Θ_t.
Initialization:
Initialize $\mathbf{W}$, $\mathbf{W_d}$, $\mathbf{b_s}$, $\mathbf{b_t}$, $\mathbf{b_d}$, Θ_c, Θ_s, Θ_t with random values
Initialize $\mathbf{\Omega} = \frac{1}{4}\mathbf{I_4}$, where $\mathbf{I}$ is an identity matrix
epoch $= 0$
while epoch $\leq$ MaxEpoch **do**
    c_s, c_t = 0, 0
    while $c_s \leq$ src_batches **do**
        read the c_sth mini-batch from the source domain
        Update Θ_c, Θ_s, $\mathbf{W}$, $\mathbf{W_d}$, $\mathbf{b_s}$ and $\mathbf{b_d}$
        c_s += 1
        if c_t == tgt_batches **then**
            c_t = 0
            Update $\mathbf{\Omega} = \frac{(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}}{\mathbf{tr}\left((\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}\right)}$
        end if
        read the c_tth mini-batch from the target domain
        Update Θ_c, Θ_t, $\mathbf{W}$, $\mathbf{W_d}$, $\mathbf{b_t}$ and $\mathbf{b_d}$
        c_t += 1
    end while
    epoch = epoch + 1
end while

Since it is a smooth function, we can easily compute its partial derivatives with respect to the eight parameters.

Updating $\mathbf{\Omega}$. After fixing the eight parameters, the optimization problem is as follows:

$$\begin{aligned} \min_{\mathbf{\Omega}} \quad & \mathbf{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) \\ \text{s.t.} \quad & \mathbf{\Omega} \geq 0, \quad \mathbf{tr}(\mathbf{\Omega}) = 1. \end{aligned}$$

As proved by [135], the above optimization problem has an analytical solution $\mathbf{\Omega} = \frac{(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}}{\mathbf{tr}\left((\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}\right)}$.

Finally, we present the whole procedure for training our full model as in Algorithm 1. Note that we only update $\mathbf{\Omega}$ when we scan all the target training instances once.

5.2.6 Implementation Details

In our implementation, we set the dimensionality of word embeddings l to 300, and initialize the lookup table $\mathbf{E}$ with the pre-trained vectors from *GloVe* [84]. Based on our preliminary exploration, for **BCNN**, the window size and the activation function are set to be 4 and ReLU; for the two convolution layers of **Pyramid**, the number of feature maps is set to be 8 and 16, the strides are set to be 1 and 3, and the kernel sizes are set to be 6×6 and 4×4; for the two max-pooling layers of **Pyramid**, the strides are set to be 4 and 2, and the pooling sizes are set to be 4×4 and 2×2. Besides, for λ_0 and λ_1, we set them as 0.05 and 0.0008; while for λ_2, λ_3, λ_4 and λ_5, we set them as 0.0004. AdaGrad [34] is used to train our model with an initial learning rate of 0.08.

5.3 Online System

As introduced in Section 5.1, our online chatbot system is based on traditional information retrieval techniques, where the goal is to obtain the nearest question in the knowledge base for a given customer question [46]. Figure 5.1 depicts the whole system architecture.

Specifically, we first build an indexing for all the questions in our knowledge base (KB) using *Apache Lucene* [5]. Next, given a query question, we employ TF-IDF ranking algorithm [113] in Lucene to compute its similarities to all the questions in the KB, and call back the top-K candidate questions. We then use a reranking algorithm to compute the similarities between the query and the K candidates, and obtain the most similar candidate. Finally we return the answer of the selected candidate to answer the query question. Note that in this chapter, we only consider formulating our question rerank module as a PI task, but one can also model it as an NLI task.

Our existing reranking method is based on this ensemble method for the An-

[5] `https://lucene.apache.org/core/`

		Train	Dev	Test
AliExpress	Q-Q Pairs	29,884/8,624	7,622/1,968	7,569/2,133
	#Query-Q	3202/2414	781/578	777/584
	#Candi-Q	9.33	9.76	9.74
	#words per Query-Q	11.71	11.73	12.04
	#words per Candi-Q	8.37	8.37	8.54
Quora	Q-Q Pairs	404,290/149,265	N.A.	N.A.

Table 5.1: Statistics of Paraphrase Identification Datasets

swer Selection task [106]. But instead of using the output of the time-consuming LSTM model, we feed another three features, namely, Word Mover's Distance [55], keywords features [106] and the cosine distance of sentence embeddings [111] to a gradient boosted regression tree (GBDT).

To combine our model with the existing ranking method, we treat the probability of being *paraphrases* predicted by our model as an additional feature, and feed all features to GBDT for reranking.

5.4 Experiments

In this section, we describe a qualitative evaluation of our proposed methods from the following perspectives: (1) From Section 5.4.1 to Section 5.4.4, we perform an **intrinsic** evaluation by utilizing a benchmark dataset and our own dataset to show the efficiency and effectiveness of our proposed base model and domain adaptation framework; (2) In Section 5.4.5, we deploy our full model into our chatbot system, and conduct an **extrinsic** evaluation to show that our full model can bring in significant improvements to our existing online chatbots.

5.4.1 Experiment Settings

Datasets: In this section, we evaluate our methods on both Paraphrase Identification (PI) and Natural Language Inference (NLI).

	SNLI	Fiction	Travel	Slate	Telephone	Government
Train	550,125	77,348	77,350	77,306	83,348	77,350
Dev	10,000	2,000	2,000	2,000	2,000	2,000
Test	10,000	2,000	2,000	2,000	2,000	2,000

Table 5.2: Statistics of the MultiNLI Dateset

For PI, we used a recently released large-scale dateset[6] by Quora as the source domain, and our E-commerce dataset as the target domain. Based on our historical data, we constructed a question answering KB, which consists of around 15,000 commonly asked QA pairs. To create labeled question pairs, we first collected all the query questions from the chat log of conversations between clients and our staff from May 22 to May 28, 2017. For each query question, we then used *Lucene* indexing to retrieve several of its similar questions, and obtained 45,075 question pairs. Finally, we asked a business analyst to annotate all the question pairs.

For NLI, we employed a large-scale multi-genre corpus [112], which contains an image captioning domain (SNLI) and another five distinct genres/domains about written and spoken English (MultiNLI)[7]. Since the number of sentence pairs in SNLI is much larger than that in the other five domains, we took SNLI as the source domain, and the others as the target domains.

Table 5.1 and Table 5.2 summarize the statistics of our datasets. Note that in Table 5.1, the number before and after the slash for Q-Q pairs denote respectively the total number of question pairs and the number of positive question pairs (i.e., paraphrases), while the two numbers for #Query-Q respectively denote the total number of query questions and the number of questions with paraphrasing candidates. Besides, for #Candi-Q, we refer to the average number of candidate questions for each query.

Compared Methods: For base models, we compared our hCNN model with the following models:

[6]https://www.kaggle.com/c/quora-question-pairs

[7]For the MultiNLI dataset, we use Version 0.9 in this chapter. Note that since the label of the original test set is unavailable, we treat its development set as our test set, and randomly choose 2000 sentence pairs from its training set as our development set.

- **BCNN** is the left component of our hCNN model, which incorporates element-wise comparisons on top of the base model proposed in [125].
- **Pyramid** is the right component of our hCNN model based on sentence interactions as in [82].
- **ABCNN** is the attention-based CNN model by [125].
- **BiLSTM** is similar to **BCNN**, but uses LSTM instead of CNN to encode each sentence as in [17].
- **ESIM** is one of the state-of-the-art attention-based LSTM models on SNLI proposed by [22].
- **hCNN** is our hybrid CNN model as introduced in Section 5.2.4.

For evaluating the proposed domain adaptation framework, we employed the following compared systems:

- **Tgt-Only** is the baseline trained in the target domain.
- **Src-Only** is another baseline trained in the source domain.
- **Mixed** is to simply combine the labeled data in the two domains to train the hCNN model.
- **Fine-Tune** is a widely used SDA method, where we first train a model on the source data, and then use the learned parameters to initialize the model parameters for training another model on the target data.
- **FS** and **SS** are the fully-shared and specific-shared frameworks as detailed in Section 5.2.2.
- **DRSS** is our proposed model of learning domain relationships based on **SS** as in Section 5.2.2.

	E-Commerce			SNLI	
	AUC	ACC	Test Time(ms)	ACC	Test Time(ms)
BCNN	81.0	77.5	2.8	81.0	3.3
Pyramid	77.8	77.0	3.8	77.7	5.3
ABCNN	81.0	78.2	5.2	81.8	12.3
hCNN	82.2†	79.2†	4.3	83.2†	6.4
BiLSTM	79.9	77.8	7.1	80.6	19.6
ESIM	84.2	79.8	32.2	86.7	79.5

Table 5.3: A Comparison Between Different Base Models

- **SS-Adv** and **DRSS-Adv** denote adding the adversarial loss into **SS** and **DRSS** as in Section 5.2.3.

All the methods in this chapter are implemented with Tensorflow and are trained on machines with NVIDIA Tesla K40m GPU.

Evaluation Metrics: For PI, since our goal is to retrieve the most similar candidate for each query question, we use our model to predict each candidate's probability of being *paraphrase* as its similarity score, and then rank all the candidates. To evaluate the ranking performance, we use Precision@1, Recall@1, F_1@1 as metrics; to evaluate the classification performance for all question pairs, we employ two metrics: the Area under the Receiver Operating Characteristic curve (AUC) score [19] and the classification accuracy (ACC). For NLI, we only use ACC as the evaluation metric.

5.4.2 Comparisons Between Base Models

In Table 5.3, we compared different models for classifying sentence pairs with hCNN in both efficiency and effectiveness. Note that to fairly evaluate the efficiency of each model, we compute the total time of predicting all the test sentence pairs on CPU by setting the mini-batch size to 1, and report the average time. Also, for feature map sizes in BCNN, ABCNN and the BCNN component in hCNN, we set them as 50 for PI and 300 for NLI.

First, we can find that LSTM-based methods are generally much slower than

	Prec@1	Rec@1	F1@1	ACC	AUC
Tgt-Only	0.717	0.551	0.623	0.792	0.822
Src-Only	0.619	0.368	0.461	0.719	0.686
Mixed	0.735	0.532	0.618	0.788	0.810
Fine-Tune	0.713	0.567	0.632	0.790	0.825
FS	0.734	0.595	0.657	0.797	0.831
SS	0.744	0.601	0.665	0.800	0.837
SS-Adv	0.743	0.603	0.666	0.808	0.842
DRSS	**0.757**	0.608	0.674†	**0.812**†	0.847
DRSS-Adv	0.753	**0.620**	**0.680**†	0.809	**0.849**

Table 5.4: The Result of Paraphrase Identification Task

CNN-based methods. Especially for ESIM, although it can outperform all CNN-based models, its computational time for each sentence pair is 32.2ms for our dataset and 79.5ms for SNLI, which is 6-11 times of CNN-based models. This means that most existing state-of-the-art models can only support low QPS, and therefore hard to be applied to industry. Second, clearly for both tasks, hCNN performs better than the other CNN-based methods, which indicates that BCNN and Pyramid are complementary to each other, and can work better when combined. Moreover, we verified that the improvements of hCNN over the other methods are significant with $p < 0.05$ based on McNemar's paired significance test [37]. Finally, while the computational cost of hCNN is slightly higher than BCNN and Pyramid, it can still serve 233 question pairs per second, which is able to satisfy the current demand of our industrial bot.

5.4.3 Comparisons Between DA Methods

We further evaluated the performance of our domain adaptation method in Table 5.4 and Table 5.5.

We can observe from Table 5.5 that for all the five target domains, Src-Only perform much worse than Tgt-Only, and the average performance of Mixed is even worse than Tgt-Only. This implies that the source domain is quite different from all the target domains, and simply mixing the training data in two domains may lead to

	Fict.	Trav.	Gov.	Tele.	Slate	AVG
Tgt-Only	0.647	0.658	0.692	0.644	0.579	0.644
Src-Only	0.520	0.516	0.540	0.520	0.488	0.517
Mixed	0.647	0.647	0.675	0.648	0.580	0.639
Fine-Tune	0.653	0.652	0.684	0.651	0.591	0.646
FS	0.662	0.671	0.704	0.657	0.588	0.656
SS	0.653	0.668	0.700	0.668	0.592	0.656
SS-Adv	0.666	0.666	0.701	0.664	0.597	0.659
DRSS	0.665	**0.674**†	0.706†	0.673†	0.605†	0.665
DRSS-Adv	**0.676**†	0.673	**0.707**†	**0.675**†	**0.607**†	**0.668**

Table 5.5: The Classification Result of NLI Task

the model overfitting the source data since $|\mathcal{D}^s|$ is much larger than $|\mathcal{D}^t|$. In addition, it is observed that the widely used Fine-Tune method can perform slightly better than Tgt-Only in most cases, which shows that pre-training the model parameters on a related source domain is better than randomly initializing them. Moreover, in all the five domains, the performance of two existing supervised domain adaptation frameworks FS and SS are both 1.9% better than that of Tgt-Only, which proves their usefulness. Furthermore, our proposed method DRSS improves the average performance of SS to 0.665, and the improvements are significant over all the tasks with $p < 0.05$ based on McNemar's paired significance test. This suggests that capturing the relationship between domains is generally useful for domain adaptation. Finally, we can see that the incorporation of adversarial loss into SS and DRSS further boosts their performance, and DRSS-Adv can achieve the best accuracy across all the methods. Similar trends can be also observed for the PI task from Table 5.4. Interestingly, by comparing Table 5.3 and Table 5.4, we find that with the help of training data from the source domain, the performance of DRSS-Adv is even better than that of ESIM. These observations demonstrate the effectiveness of our domain adaptation method.

Apart from the effectiveness, we also measure the efficiency of each method. Since the first five methods only use a single hCNN model for prediction, the computational time is the same as hCNN. As for SS, DRSS and their adversarial exten-

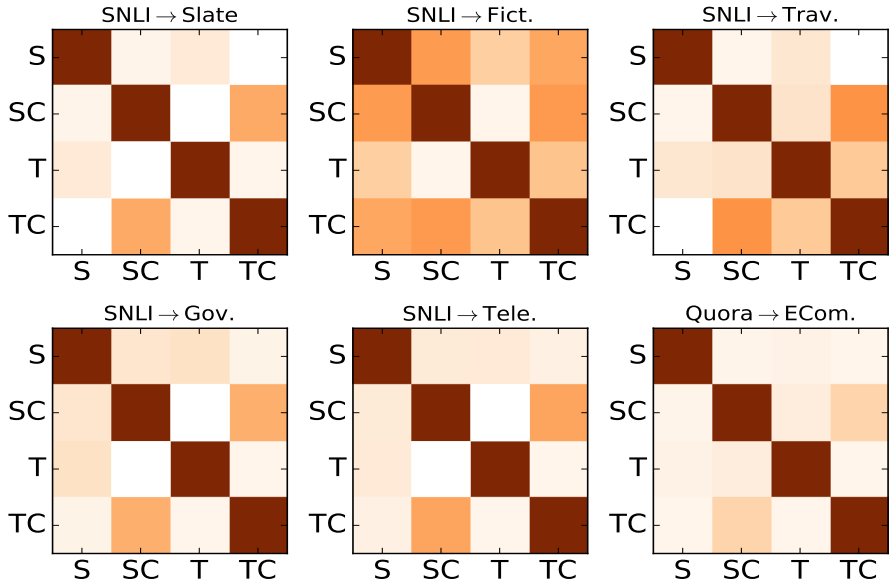

Figure 5.4: Learnt Correlation Matrix. A darker color means a larger entry value. S:Source, T:Target, SC:Source-shared, TC: Target-shared.

sions, the computational time is 6.9ms, which is slightly longer than the other five methods but still much shorter than LSTM-based methods as in Table 5.3.

5.4.4 Domain Relationships

After obtaining the covariance matrix Ω for each source/target pair, we can derive their corresponding correlation matrices. For better comparison, here we show the square root of the correlation matrices for DRSS.

As shown in Figure 5.4 that across all the six source/target pairs, $\mathbf{W_{sc}}$ and $\mathbf{W_{tc}}$ are positively related with each other. This is intuitive as the shared network is supposed to learn shared features between the source and the target domains, thus the learned $\mathbf{W_{sc}}$ and $\mathbf{W_{tc}}$ should be close to each other. This also shows the learned correlation matrix helps to capture the inter-domain relationship between $\mathbf{W_{sc}}$ and $\mathbf{W_{tc}}$.

In Figure5.4, we can also see that for most source/target pairs except SNLI→Fict, the correlation between $\mathbf{W_s}$ and $\mathbf{W_{sc}}$ and that between $\mathbf{W_t}$ and $\mathbf{W_{tc}}$ learnt by our model are with small values. This indicates that in most cases, the shared feature space and the domain-specific feature space learnt by SS tend to be

different from each other, and our model can help to reveal such intra-domain relationships.

Finally, to help us get a deeper insight on the helpfulness of the adversarial training, we perform comparisons on the correlation matrices learnt by DRSS and DRSS-Adv. We first show the result of SNLI→Fict. in Table 5.6. As we can see, for DRSS, the correlation between $\mathbf{W_s}$ (or $\mathbf{W_t}$) and $\mathbf{W_{sc}}$ (or $\mathbf{W_{tc}}$) is relatively large, while for DRSS-Adv, the correlation is relatively small. For the other subtasks, we find that the learnt matrices of DRSS-Adv are similar to those of DRSS, but we still observe that the intra-domain correlations of DRSS-Adv are generally smaller than those of DRSS. This shows that adding the adversarial loss can encourage the shared feature space to capture more domain-independent features, and further make the shared and domain-specific feature spaces more different. Therefore, the adversarial training can lead our model to better satisfy our assumption on the domain relationships, and finally improve the performance. All the above observations demonstrate that our model can capture the *inter-domain* and *intra-domain* relationship as mentioned in Section 5.2.2.

5.4.5 Extrinsic Evaluations

As mentioned in Section 5.3, for the online reranking algorithm, we propose to train GBDT by treating the prediction score of our DRSS model as another feature. To achieve this, we first took out the prediction scores of our DRSS model on the validation set. Then, we combined them together the other features as introduced

	DRSS				DRSS-Adv			
	$\mathbf{W_s}$	$\mathbf{W_{sc}}$	$\mathbf{W_t}$	$\mathbf{W_{tc}}$	$\mathbf{W_s}$	$\mathbf{W_{sc}}$	$\mathbf{W_t}$	$\mathbf{W_{tc}}$
$\mathbf{W_s}$	1.000	0.242	0.101	0.205	1.000	0.090	0.055	0.062
$\mathbf{W_{sc}}$	0.242	1.000	0.008	0.247	0.090	1.000	0.024	0.221
$\mathbf{W_t}$	0.101	0.008	1.000	0.127	0.055	0.024	1.000	0.043
$\mathbf{W_{tc}}$	0.205	0.247	0.127	1.000	0.062	0.221	0.043	1.000

Table 5.6: Correlation Matrices on SNLI→Fict.

	Offline		Online Evaluation
	F_1@1	Time(ms per query)	Prec@1
GBDT	0.539	20.1	0.614
GBDT-DRSS	**0.681**	80.7	**0.729**

Table 5.7: The Performance of Online Serving.

in Section 5.3, and trained GBDT on the validation set. The model performance on the test set is reported in Table 5.7. Note that the test time here denotes the average serving time (including the Response Time), which is different from the reported test time in Table 5.3.

As we can see from the offline test, the GBDT model with the feature derived from our DRSS model (referred to as GBDT-DRSS) is respectively 26.3% and 7.1% better than our existing online model (referred to as GBDT) and the GBDT model with the feature derived from hCNN (refered to as GBDT-hCNN) in F_1@1. Although adding our DRSS feature leads to more computational time, the total prediction time is 80.7ms for each query question (i.e., QPS of 12), which is acceptable for our chatbots.

For online serving, to accelerate the computation, we set the number of candidates returned by *Lucene* as 30, and bundle the 30 candidates into a mini-batch to feed into our model for prediction. For online evaluation, we randomly sampled 2750 questions, where 1317 questions are answered by GBDT and 1433 questions are answered by GBDT-DRSS. Then, we asked one business analyst to annotate if the nearest question returned by models expresses the same meaning as the query question, and compared their precision at top-1. As shown in Table 5.7, the Prec@1 of GBDT-DRSS is 18.8% higher than that of GBDT.

5.5 Discussion

In this chapter, we systematically evaluated different base methods and domain adaptation techniques for modelling sentence pairs, with the goal of proposing an

effective and efficient domain adaptation framework for PI and NLI. Specifically, we first proposed a hybrid CNN model on the basis of two existing models, and then further proposed a general supervised domain adaptation framework, which can simultaneously perform the shared feature learning and domain relationship learning in an end-to-end mode. Evaluations on both a benchmark dataset and our own dataset showed that (1) our hybrid CNN model is both effective and efficient in comparison with several representative models; (2) our domain adaptation framework can outperform all the existing frameworks across six source/target pairs. We further deployed our full domain adaptation model in our online chatbot system, and showed that it can improve the performance of the existing system by a large margin.

Chapter 6

Improving Multi-label Emotion Classification via Sentiment Classification with Dual Attention Transfer Network

In this chapter, we target at improving the performance of multi-label emotion classification with the help of sentiment classification. Specifically, we propose a new transfer learning architecture to divide the sentence representation into two different feature spaces, which are expected to respectively capture the general sentiment words and the other important emotion-specific words via a dual attention mechanism. Extensive experimental results demonstrate that our transfer learning approach can outperform several strong baselines and achieve the state-of-the-art performance on two benchmark datasets.

6.1 Introduction

In recent years, the number of user-generated comments on social media platforms has grown exponentially. In particular, social platforms such as Twitter allow

ID	**Tweet**	**Emotion**
T1	AI revolution, soon is possible #fearless #good #goodness	joy, optimism
T2	Shitty is the worst feeling ever #depressed #anxiety	fear, sadness
T3	I am back lol. #revenge	joy, anger

Table 6.1: Example Tweets from SemEval-18 Task 1.

users to easily share their personal opinions, attitudes and emotions about any topic through short posts. Understanding people's emotions expressed in these short posts can facilitate many important downstream applications such as emotional chatbots [139], personalized recommendations, stock market prediction, policy studies, etc. Therefore, it is crucial to develop effective emotion detection models to automatically identify emotions from these online posts.

In the literature, emotion detection is typically modeled as a supervised multi-label classification problem, because each sentence may contain one or more emotions from a standard emotion set containing *anger*, *anticipation*, *disgust*, *fear*, *joy*, *love*, *optimism*, *pessimism*, *sadness*, *surprise* and *trust*. Table 6.1 shows three example sentences along with their emotion labels. Traditional approaches to emotion detection include lexicon-based methods [110], graphical model-based methods [61] and linear classifier-based methods [89, 60]. Given the recent success of deep learning models, various neural network models and advanced attention mechanisms have been proposed for this task and have achieved highly competitive results on several benchmark datasets [109, 1, 35, 7, 39, 52].

However, these deep models must overcome a heavy reliance on large amounts of annotated data in order to learn a robust feature representation for multi-label emotion classification. In reality, large-scale datasets are usually not readily available and costly to obtain, partly due to the ambiguity of many informal expressions in user-generated comments. Conversely, it is easier to find datasets (especially in English) associated with another closely-related task: sentiment classification, which aims to classify the sentiment polarity of a given piece of text (i.e., positive, negative and neutral). We expect that these resources may allow us to improve sentiment-sensitive representations and thus more accurately identify emotions in

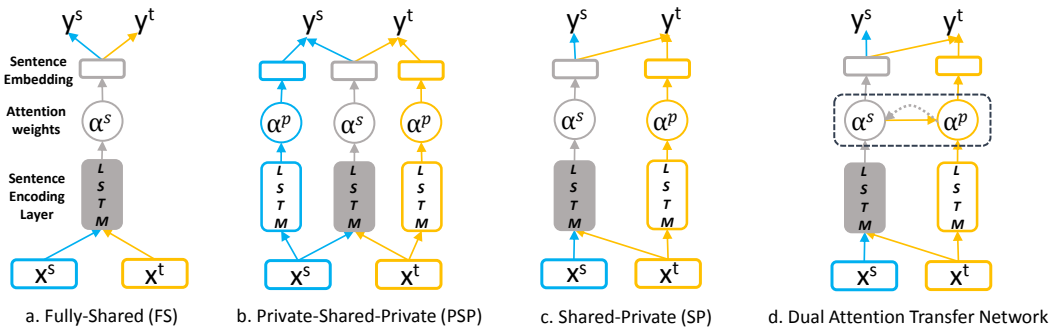

Figure 6.1: Overview of Different Transfer Learning Models.

social media posts. To achieve these goals, we propose an effective transfer learning (TL) approach in this chapter.

Most existing TL methods either 1) assume that both the source and the target tasks share the same sentence representation [70] or 2) divide the representation of each sentence into a shared feature space and two task-specific feature spaces [64, 133], as demonstrated by Figure 6.1.a and Figure 6.1.b. However, when applying these TL approaches to our scenario, the former approach may lead the learnt sentence representation to pay more attention to general sentiment words such as *good* but less attention to the other sentiment-ambiguous words like *shock* that are also integral to emotion classification. The latter approach can capture both the sentiment and the emotion-specific words. However, some sentiment words only occur in the source sentiment classification task. These words tend to receive more attention in the source-specific feature space but less attention in the shared feature space, so they will be ignored in our emotion classification task. Intuitively, any sentiment word also indicates emotion and should not be ignored by our emotion classification task.

Therefore, we propose a shared-private (SP) model as shown in Figure 6.1.c, where we employ a shared LSTM layer to extract shared sentiment features for both sentiment and emotion classification tasks, and a target-specific LSTM layer to extract specific emotion features that are only sensitive to our emotion classification task. However, as pointed out by [64] and [133], it is not guaranteed that such a simple model can well differentiate the two feature spaces to extract shared and

target-specific features as we expect. Take the sentence **T1** in Table 6.1 as an example. Both the shared and task-specific layers could assign higher attention weights to *good* and *goodness* due to their high frequencies in the training data but lower attention weights to *fearless* due to its rare occurrences. In this case, this SP model can only predict the *joy* emotion but ignores the *optimism* emotion. Hence, to enforce the orthogonality of the two feature spaces, we further introduce a dual attention mechanism, which feeds the attention weights in one feature space as extra inputs to compute those in the other feature space, and explicitly minimizes the similarity between the two sets of attention weights. Experimental results show that our dual attention transfer architecture can bring consistent performance gains in comparison with several existing transfer learning approaches, achieving the state-of-the-art performance on two benchmark datasets.

6.2 Methodology

6.2.1 Base Model for Emotion Classification

Given an input sentence, the goal of emotion analysis is to identify one or multiple emotions contained in it. Formally, let $\mathbf{x} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ be the input sentence with n words, where $\mathbf{w}_j$ is a d-dimensional word vector for word w_j in the vocabulary $\mathcal{V}$, and is retrieved from a lookup table $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$. Moreover, let $\mathcal{E}$ be a set of pre-defined emotion labels. Accordingly, for each $\mathbf{x}$, our task is to predict whether it contains one or more emotions in $\mathcal{E}$. We denote the output as $\mathbf{e} \in \{0, 1\}^K$ where $\mathbf{e}_k \in \{0, 1\}$ denotes whether or not $\mathbf{x}$ contains the k-th emotion. We further assume that we have a set of labeled sentences, denoted by $D^{\text{e}} = \{\mathbf{x}^{(i)}, \mathbf{e}^{(i)}\}_{i=1}^{N}$.

Sentence Representation: As any standard text classification problem, the hidden representation $\mathbf{h}_j$ of each word $\mathbf{w}_j$ in the input sentence can be computed with any effective neural network architectures, including CNN [50], RNN [24] and Tree-Structured Neural Networks (TNN) [96, 100]. Since our goal is to leverage

sentiment classification to help improve the state-of-the-art performance of multi-label emotion classification, we use the attention-based LSTM model as our base model in this Chapter, which has been shown to perform much better than CNN-based approaches [39, 7]. Specifically, we first employ the standard bi-directional Long Short Term Memory (Bi-LSTM) network to sequentially process each word in the input:

$$\overrightarrow{\mathbf{h}_j} = \text{LSTM}(\overrightarrow{\mathbf{h}_{j-1}}, \mathbf{x}_j, \Theta_f),$$
$$\overleftarrow{\mathbf{h}_j} = \text{LSTM}(\overleftarrow{\mathbf{h}_{j+1}}, \mathbf{x}_j, \Theta_b),$$

where Θ_f and Θ_b denotes all the parameters in the forward and backward LSTM. Then, for each word $\mathbf{x}_j$, its hidden state $\mathbf{h}_j \in \mathbb{R}^d$ is generated by concatenating $\overrightarrow{\mathbf{h}_j}$ and $\overleftarrow{\mathbf{h}_j}$ as $\mathbf{h}_j = [\overrightarrow{\mathbf{h}_j}; \overleftarrow{\mathbf{h}_j}]$.

For emotion classification, since emotion words are relatively more important for final predictions, we adopt the widely used attention mechanism [6] to select the key words for sentence representation. Specifically, we first take the final hidden state $\mathbf{h}_n$ as a sentence summary vector $\mathbf{z}$, and then obtain the attention weight α_i for each hidden state $\mathbf{h}_j$ as follows:

$$u_j = \mathbf{v}^\top \tanh(\mathbf{W}_h \mathbf{h}_j + \mathbf{W}_z \mathbf{z}), \tag{6.1}$$
$$\alpha_j = \frac{\exp(u_j)}{\sum_{l=1}^{n} \exp(u_l)}, \tag{6.2}$$

where $\mathbf{W}_h, \mathbf{W}_z \in \mathbb{R}^{a \times d}$ and $\mathbf{v} \in \mathbb{R}^a$ are learnable parameters. The final sentence representation $\mathbf{H}$ is computed as:

$$\mathbf{H} = \sum_{j=1}^{n} \alpha_j \mathbf{h}_j.$$

Output Layer: We first apply a Multilayer Perceptron (MLP) with one hidden layer on top of $\mathbf{H}$, followed by normalizing it to obtain the probability distribution over

all of the emotion labels:

$$p(\mathbf{e}^{(i)} \mid \mathbf{H}) = \mathbf{o}^{(i)} = \text{softmax}(\text{MLP}(\mathbf{H})).$$

Then, we propose to minimize the KL divergence between our predicted probability distribution and the normalized ground truth distribution as our objective function:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{e}_k^{(i)} \big(\log(\mathbf{e}_k^{(i)}) - \log(\mathbf{o}_k^{(i)}) \big).$$

During the test stage, we will select a threshold γ on the development set so that the emotion with scores higher than γ will be predicted as 1.

6.2.2 Transfer Learning Architecture

Due to the limited number of annotated data for multi-label emotion classification, here we resort to sentiment classification to consider a transfer learning scenario. Let $D^{\text{s}} = \{\mathbf{x}^{(m)}, y^{(m)}\}_{m=1}^{M}$ be another set of labeled sentences for sentiment classification, where $y^{(m)}$ is the ground-truth label indicating whether the m-th sentence is *positive*, *negative* or *neutral*.

Shared-Private (SP) Model

Intuitively, sentiment classification is a coarse-grained emotion analysis task, and can be fully leveraged to learn a more robust sentiment-sensitive representation. Therefore, we first use a shared attention-based Bi-LSTM layer to transform the input sentences in both tasks into a shared hidden representation $\mathbf{H_c}$, and also employ another task-specific Bi-LSTM layer to get the target-specific hidden representation $\mathbf{H_t}$. Next, we employ the following operations to map the hidden representations to

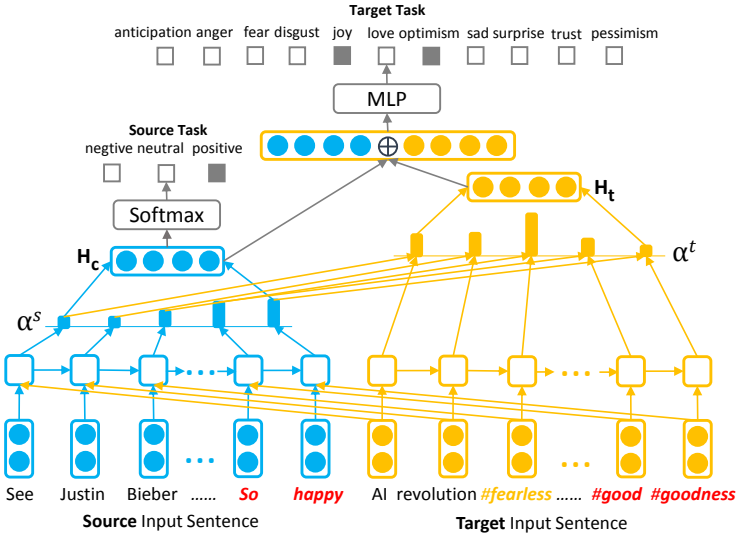

Figure 6.2: Dual Attention Transfer Network.

the sentiment label y and the emotion label $\mathbf{e}$:

$$p(y^{(m)}|\mathbf{H_c}) = \text{softmax}\big(\mathbf{W}^s\mathbf{H_c} + \mathbf{b}^s\big),$$
$$p(\mathbf{e}^{(i)}|\mathbf{H_c}, \mathbf{H_t}) = \text{softmax}\big(\text{MLP}([\mathbf{H_c}; \mathbf{H_t}])\big),$$

where $\mathbf{W}^s \in \mathbb{R}^{d\times 3}$ and $\mathbf{b}^s \in \mathbb{R}^3$ are the parameters for the source sentiment classification task.

Proposed Dual Attention Transfer Network (DATN)

As we introduced before, the shared and target-specific feature spaces in the above SP model are expected to respectively capture the general sentiment words and the task-specific emotion words. However, without any constraint, the two feature spaces may both tend to pay more attention to frequently occurring and important sentiment words like *great* and *happy*, but less to those rarely occurring but crucial emotion words like *anxiety* and *panic*. Therefore, to encourage the two feature spaces to focus on sentiment words and emotion-specific words respectively, we propose using the attention weights computed from the shared layer as extra inputs to compute the attention weights of the target-specific layer. Specifically, as shown in Figure 6.2, we first use Eq.6.1 and Eq.6.2 to compute the attention weights α^s in

the shared layer, and then use the following equation to obtain the attention weights α^t in the target specific layer:

$$u_j^t = \mathbf{v}^{t^\top} \tanh(\mathbf{W}_h^t \mathbf{h}_j^t + w_\alpha \alpha_j^s + \mathbf{W}_z^t \mathbf{z}^t),$$
$$\alpha_j^t = \frac{\exp(u_j^t)}{\sum_{l=1}^{n} \exp(u_l^t)}.$$

In addition, we introduce another similarity loss to explicitly enforce the difference between the two attention weights and minimize the cosine similarity between α^s and α^t.

Finally, our combined objective function is defined as follows:

$$\mathcal{J} = -\frac{1}{M} \sum_{m=1}^{M} \log p(y_m | \mathbf{H_c}) + \mathcal{L} + \lambda \sum_{i=1}^{N} \text{cos\_sim}(\alpha_i^s, \alpha_i^t),$$

where λ is a hyperparameter used to control the effect of the similarity loss.

Model Details

During the training stage, we adopted the widely used alternating optimization strategy, which iteratively samples one mini-batch from D^{s} for only updating the parameters in the left part of our model, followed by sampling another mini-batch from D^{e} for updating all the parameters in our model. It is also worth noting that in Figure 6.2, we first obtain the shared attention weights α^s and feed it as extra inputs to compute α^t. In fact, to differentiate the attention weights in the two feature spaces, we can also first compute α^t, followed by computing α^s based on α^t. We refer to these two variants of our model as DATN-1 and DATN-2 respectively.

6.3 Experiments

6.3.1 Experiment Settings

Datasets: We conduct experiments on both English and Chinese languages.

	Dataset	Train	Dev	Test	Words
E1	SemEval-18	6,838	886	3,259	32,557
S1	SemEval-16	28,631	-	-	40439
E2	Ren-CECps-1	13,841	1,972	3,602	40,099
S2	Ren-CECps-2	15,199	-	-	

Table 6.2: The number of sentences in each dataset.

For **English**, we employ a widely used Twitter dataset from SemEval 2016 Task 4A [72] as our source sentiment classification task. For our target emotion classification task, we use the Twitter dataset recently released by SemEval 2018 Task 1C [68], which contains 11 emotions as shown in the top of Figure 6.2. To tokenize the tweets in our dataset, we follow [75] by adopting most of their preprocessing rules except that we split the hashtag into '#' and its subsequent word.

For **Chinese**, we use a well-known Chinese blog dataset Ren-CECps from [88], which contains 1487 documents with each sentence labeled by a sentiment label and 8 emotion labels: *anger*, *expectation*, *anxiety*, *joy*, *love*, *hate*, *sorrow* and *surprise*. Given the difficulty of finding a large-scale sentiment classification dataset specific to Chinese blogs, we simply divided the original dataset to form our source and target tasks[1]. The basic statistics of our two datasets are summarized in Table 6.2.

Parameter Settings: The word embedding size d is set to be 300 for E1 and 200 for E2, and the lookup table $\mathbf{E}$ is initialized by pre-trained word embeddings based on Glove[2]. The hidden dimension and the number of LSTM layers in both datasets are set to be 200 and 1. During training, Adam [53] is used to schedule the learning rate, where the initial learning rate is set to be 0.001. Also, the dropout rate is set to 0.5. After tuning, λ is set as 0.05 for both datasets, and γ is set as 0.12 for E1 and 0.2 for E2. All the models are implemented with Tensorflow.

Evaluation Metrics: We take the official code from SemEval-18 Task 1C and use accuracy and Macro F1 score as main metrics. For E2, we follow [136] to use average precision (AP) and one error (OE) as secondary metrics.

[1]The first 560/80/160 documents are used as train/dev/test set for emotion classification, and the remaining 687 documents are used for sentiment classification.

[2]https://nlp.stanford.edu/projects/glove/.

Methods	S1 → E1		S2 → E2			
	ACC↑	**F1↑**	**ACC↑**	**F1↑**	**AP↑**	**OE↓**
Base	0.569	0.521	0.368	0.399	0.648	0.531
FT	0.575	0.519	0.372	0.398	0.655	0.519
FS	0.577	0.526	0.386	0.403	0.662	0.507
PSP	0.579	0.531	0.384	0.405	0.658	0.517
APSP	0.580	0.540	0.389	0.399	0.670	0.499
SP	0.577	0.532	0.389	0.410	0.667	0.507
DATN-1	0.582	0.543	0.393	0.410	0.670	0.501
DATN-2	**0.583**	**0.544**	**0.400**	**0.420**	**0.674**	**0.498**
Rank 2	0.582	0.534	-	0.392	0.641	0.523
Rank 1	0.595	0.542	-	0.416	0.680	0.455
DATN-2*	***0.597***	***0.551***	-	-	-	-
Base†	-	-	0.445	0.426	0.725	0.425
DATN-2†	-	-	***0.457***	***0.444***	***0.732***	***0.415***

Table 6.3: The results of different transfer learning methods by averaging ten runs (top) and the comparison between our best model and the state-of-the-art systems (bottom). DATN-2* indicates the ensemble results of ten runs. Base† and DATN-2† denotes the average results of conducting ten-fold cross validation on the whole dataset for fair comparison, and here for the source and target tasks in DATN-2†, we use the same training data. For E1, Rank1 and Rank2 are the top two systems from the official leadboard; For E2, Rank1 and Rank2 are from [137, 136].

6.3.2 Results

To better evaluate our proposed methods, we employed the following systems for comparison: 1) *Base*, training our base model in Section 6.2.1 only on D^{e}; 2) *FT* (Fine-Tuning), using D^{s} to pre-train the whole model, followed by using D^{e} to Fine Tune the model parameters; 3) *FS*, the Fully-Shared framework by [70] as shown in Figure 6.1.a; 4) *PSP* and *APSP*, the Private-Shared-Private framework and its extension with Adversarial losses by [64] as shown in Figure 6.1.b; 5) *SP*, *DATN-1* and *DATN-2*, the Shared-Private model and two variants of our Dual Attention Transfer Network as shown in Figure 6.1.c and Figure 6.1.d.

In Table 6.3, we report the comparison results between our method and the baseline systems. It can be easily observed that 1) for transfer learning, although the performance of *SP* is similar to or even lower than some baseline systems, our proposed dual attention models, i.e., *DATN-1* and *DATN-2*, can generally boost *SP* to achieve the best results. To investigate the significance of the improvements, we combine

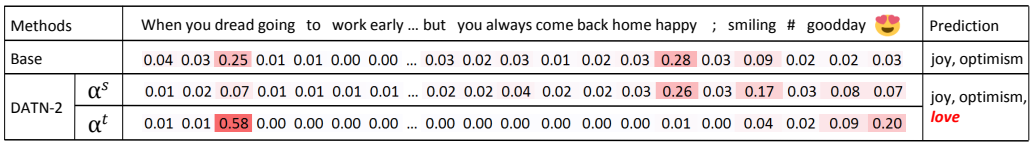

Figure 6.3: Comparison of attention weights between *Base* and our *DATN-2* model on a test sentence from SemEval-18. Note that the ground truth emotion labels for this example are *joy*, *optimism* and *love*.

each model's predictions of all emotion labels followed by treating them as a single label, and then perform McNemar's significance tests [37]. Finally, we verify that for English, *DATN-1* is significantly better than *Base*, *FT*, *FS* and *SP*, while *DATN-2* is significant better than all the methods except *APSP*; for Chinese, *DATN-1* and *DATN-2* are significantly better than all the compared methods. 2) Even compared with the state-of-the-art systems in E1 which also employ other external resources, including the affective embedding, emotion lexicon and sentiment classification datasets [7], the ensemble results of *DATN-2* can achieve slightly better performance; in addition, it is clear that our model can obtain the best performance in E2.

Furthermore, to obtain a better understanding of the advantage of our method, we choose one sentence from the test set of E1, and visualize the attention weights obtained by *Base* and *DATN-2* in Figure 6.3. We can see that *Base* pays more attention to those frequent emotion words while ignoring the less frequent but important emoji, and thus fails to predict the *love* emotion implied by the emoji. In contrast, with the proposed dual attention mechanism, *DATN-2* makes correct predictions since it can respectively capture the general sentiment words and the emotion-specific emojis.

6.4 Discussion

In this chapter, we proposed a novel dual attention-based transfer learning approach to leverage sentiment classification to improve the performance of multi-label emotion classification. The model divides the sentence representation into shared and

target-specific feature spaces, and utilizes a dual attention mechanism to enforce the two feature spaces to capture different key information in the sentence. Using two benchmark datasets, we show the effectiveness of the proposed transfer learning method.

Chapter 7

Conclusion

7.1 Summary

Although standard supervised learning, especially deep learning, has been extensively applied in a myriad of NLP tasks in recent years, the heavy reliance on annotated data hinders its performance in those resource-scarce domains or tasks. Therefore, in this dissertation, we focus on proposing effective transfer learning approaches to leverage the annotations in other resource-rich domains/tasks to improve the model performance in our target domain/task under two different settings: *unsupervised transfer learning* and *supervised transfer learning*.

Specifically, this thesis makes the following main contributions to transfer learning in NLP:

- **Unsupervised transfer learning**

  First, we argue that most existing domain adaptation methods are unscalable and computationally costly due to their reliance on dimensionality reduction and multi-layer auto-encoders. To address this issue, we propose a hassle-free domain adaptation method to derive the domain-independent feature representations for each instance by simply calculating its similarity with a set of *exemplar* vectors chosen from the target domain. **Moreover**, we further explore how to extend a famous domain adaptation method (SCL) to neural net-

work models, and propose a general auxiliary task-based neural domain adaptation framework to help induce the shared representations across domains. Besides, to apply the general framework to sentiment classification, we also carefully design several domain-independent auxiliary tasks, and demonstrate the significant improvements of our method over existing domain adaptation methods in both sentence-level and document-level sentiment classification.

- **Supervised transfer learning**

  Since the core idea of transfer learning is to only transfer the shared knowledge while exclude the source-specific knowledge from the target domain/-task, we aim to develop different techniques to better characterize the shared knowledge and domain/task-specific knowledge in previous transfer learning approaches. To achieve this goal, we **first** propose a unified model to explicitly model the inter-domain and intra-domain relationships based on an existing transfer learning framework, and then incorporate an adversarial loss to make the shared representations more invariant across domains. **Furthermore**, we propose another novel dual attention transfer network, where we incorporate several constraints into our model to enforce the shared LSTM channel to capture the shared representations and the task-specific LSTM channel to capture the task-specific representations respectively.

7.2 Future Direction

Although we proposed several transfer learning models including unsupervised transfer learning methods and supervised transfer learning methods for a number of NLP tasks in this dissertation, we acknowledge that there are still many open problems worth further investigations from our research communities in the future:

- **New Transfer Learning Models for Text Generation**

  To the best of our knowledge, in NLP, most existing transfer learning models,

including the approaches from Chapter 3 to Chapter 6, only focus on the input encoder, i.e., how to encode each instance into a shared feature space [78]. However, we argue that these transfer learning approaches may not be quite effective for many text generation tasks such as text summarization and dialogue generation, since the most popular neural architectures for text generation (i.e., sequence to sequence models) contain both a sequence encoder and a sequence decoder. Therefore, in the future, it should be a promising direction to work on cross-domain text generation tasks with the goal of proposing effective transfer learning models from the encoder and decoder perspectives.

- **Learning Shared Representations at Word Level**

  The core idea of our proposed four approaches in this thesis is to learn the shared representations for each input sentence or document. Another possible direction, which is still not well studied in NLP, is to learn cross-domain word embeddings so that the word vectors of the corresponding words in different domains are close to each other. For example, in sentiment classification, assuming that the source and the target domains are respectively *Movie* and *Restaurant*, we may expect their domain-specifc aspect words (e.g., *plot* from *Movie* domain and *ambiance* from *Restaurant* domain) and their domain-specifc sentiment sentiment words (e.g., *attractive* from *Movie* domain and *delicious* from *Restaurant* domain) have similar word embeddings. To achieve this, incorporating both the domain-invariant pivot words and the domain-independent syntactic patterns might be one possible solution to learn robust word embeddings against domain shift, and should be worthwhile to be further explored.

- **Transfer Learning with Multiple Sources**

  In this thesis, we have shown the effectiveness of leveraging a single source domain to improve the model performance in the target domain. However, in real scenarios, we may have access to a large amount of annotated data

from multiple source domains. In this case, since the similarity between each source domain and the target domain might be different from each other, we may encounter other challenges, i.e., how to define the similarity between two domains and how to better leverage the annotated data from the source domains that are more similar to the target domain. Although multi-source domain adaptation has been extensively studied in many existing work, most of them only focus on traditional discrete representations and linear classifiers, and the exploration in neural network approaches is still quite limited. Hence, an end-to-end unified neural framework is required to solve all the abovementioned challenges for multi-source domain adaptation in the future.

Bibliography

[1] M. Abdul-Mageed and L. Ungar. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[2] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853, 2005.

[3] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of recent advances in natural language processing (RANLP)*, 2005.

[4] A. Axelrod, X. He, and J. Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

[5] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, 2010.

[6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] C. Baziotis, N. Athanasiou, A. Chronopoulou, A. Kolovou, G. Paraskevopoulos, N. Ellinas, S. Narayanan, and A. Potamianos. NTUA-SLP at SemEval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. *arXiv preprint arXiv:1804.06658*, 2018.

[8] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[9] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(Sep):2137–2155, 2009.

[10] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[11] J. Blitzer, S. Kakade, and D. P. Foster. Domain adaptation with coupled subspaces. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[12] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 2006.

[13] D. Bollegala, T. Maehara, and K.-i. Kawarabayashi. Unsupervised cross-domain word representation learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

[14] D. Bollegala, T. Mu, and J. Goulermas. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on knowledge and data engineering*, 6(2):398–410, 2016.

[15] D. Bollegala, D. Weir, and J. Carroll. 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*, 2011.

[16] A. Bordes, N. Usunier, S. Chopra, and J. Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[17] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[18] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association of Computational Linguistics*, 2016.

[19] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[20] Y. S. Chan and H. T. Ng. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[21] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[22] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association of Computational Linguistics*, 2017.

[23] Y. Choi and C. Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[25] L. Cui, S. Huang, F. Wei, C. Tan, C. Duan, and M. Zhou. Superagent: A customer service chatbot for e-commerce websites. In *Proceedings of the 55th Annual Meeting of the Association of Computational Linguistics, Demonstration*, 2017.

[26] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2007.

[27] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200, 2007.

[28] H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[29] H. Daumé III and J. Jagarlamudi. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[30] H. Daumé III, A. Kumar, and A. Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, 2010.

[31] Y. Ding, J. Yu, and J. Jiang. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[32] L. Dong, F. Wei, K. Xu, S. Liu, and M. Zhou. Adaptive multi-compositionality for recursive neural network models. *IEEE Transactions on Audio, Speech & Language Processing*, 24(3):422–431, 2016.

[33] M. Dredze and K. Crammer. Online methods for multi-domain learning and adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[34] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[35] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[36] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[37] L. Gillick and S. J. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *International Conference onAcoustics, Speech, and Signal Processing*, 1989.

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

[39] H. He and R. Xia. Joint binary neural network for multi-label learning with applications to emotion classification. *arXiv preprint arXiv:1802.00891*, 2018.

[40] L. Hirschman, M. Colosimo, A. Morgan, and A. Yeh. Overview of BioCreAtIvE task 1B: normailized gene lists. *BMC Bioinformatics*, 6(Suppl 1):S11, 2005.

[41] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, pages 1735–1780, 1997.

[42] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, 2014.

[43] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*, 2004.

[44] Y. Hu, K. Zhai, V. Eidelman, and J. Boyd-Graber. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014.

[45] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[46] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2005.

[47] J. Jiang and C. Zhai. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[48] T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009.

[49] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 7:881–892, 2002.

[50] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[51] Y. Kim, C. Denton, L. Hoang, and A. M. Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[52] Y. Kim, H. Lee, and K. Jung. Attnconvnet at semeval-2018 task 1: Attention-based convolutional neural networks for multi-label emotion classification. *arXiv preprint arXiv:1804.00831*, 2018.

[53] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[54] A. Kumar, A. Saha, and H. Daume. Co-regularization based semi-supervised domain adaptation. In *Advances in neural information processing systems*, 2010.

[55] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *international Conference on Machine Learning*, 2015.

[56] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, 2016.

[57] T. Lei, R. Barzilay, and T. Jaakkola. Molding CNNs for text: non-linear, non-consecutive convolutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

[58] F. Li, S. J. Pan, O. Jin, Q. Yang, and X. Zhu. 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*, 2012.

[59] F.-L. Li, M. Qiu, H. Chen, X. Wang, X. Gao, J. Huang, J. Ren, Z. Zhao, W. Zhao, L. Wang, and G. Jin. Alime assist: An intelligent assistant for creating an innovative e-commerce experience. In *Proceedings of the ACM International on Conference on Information and Knowledge Management. Demonstration*, 2017.

[60] L. Li, H. Wang, X. Sun, B. Chang, S. Zhao, and L. Sha. Multi-label text categorization with joint learning predictions-as-features method. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[61] S. Li, L. Huang, R. Wang, and G. Zhou. Sentence-level emotion classification with label and context dependence. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

[62] Z. Li, Z. Yu, W. Ying, W. Yuxiang, and Q. Yang. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017.

[63] B. Liu, M. Huang, J. Sun, and X. Zhu. Incorporating domain and sentiment supervision in representation learning for domain adaptation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[64] P. Liu, X. Qiu, and X. Huang. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[65] M. Long, J. Wang, G. Ding, S. J. Pan, and S. Y. Philip. Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1076–1089, 2014.

[66] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[67] M. Miwa and M. Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[68] S. M. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko. Semeval-2018 task 1: Affect in tweets. In *SemEval*, 2018.

[69] L. Mou, R. Men, G. Li, Y. Xu, L. Zhang, R. Yan, and Z. Jin. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association of Computational Linguistics*, 2016.

[70] L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[71] T. Nakagawa, K. Inui, and S. Kurohashi. Dependency tree-based sentiment classification using CRFs with hidden variables. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2010.

[72] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. In *SemEval*, 2016.

[73] M. L. Nguyen, I. W. Tsang, K. M. A. Chai, and H. L. Chieu. Robust domain adaptation for relation extraction via clustering consistency. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014.

[74] T. H. Nguyen and R. Grishman. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014.

[75] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*, 2013.

[76] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of of the 19th international conference on World Wide Web*, 2010.

[77] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *Neural Networks, IEEE Transactions on*, 22(2):199–210, 2011.

[78] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[79] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 2004.

[80] B. Pang and L. Lee. 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*, 2005.

[81] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

[82] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[83] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

[84] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[85] B. Plank and A. Moschitti. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013.

[86] N. Ponomareva and M. Thelwall. 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*, 2012.

[87] M. Qiu, F.-L. Li, S. Wang, X. Gao, Y. Chen, W. Zhao, H. Chen, J. Huang, and W. Chu. Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association of Computational Linguistics*, 2017.

[88] C. Quan and F. Ren. Sentence emotion analysis and recognition based on emotion words using ren-cecps. *International Journal of Advanced Intelligence*, 2010.

[89] X. Quan, Q. Wang, Y. Zhang, L. Si, and L. Wenyin. Latent discriminative models for social emotion detection with emotional dependency. *ACM Transactions on Information Systems (TOIS)*, 34(1):2, 2015.

[90] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[91] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, 2010.

[92] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. Reasoning about entailment with neural attention. *International Conference on Learning Representations*, 2016.

[93] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[94] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, 2011.

[95] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

[96] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[97] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440, 2008.

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

[99] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

[100] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

[101] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *International Conference on Learning Representations*, 2017.

[102] M. Tan, C. d. Santos, B. Xiang, and B. Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[103] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

[104] I. Titov. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 2011.

[105] O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[106] D. Wang and E. Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association of Computational Linguistics*, 2015.

[107] D. Wang and T. F. Zheng. Transfer learning for speech and language processing. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*, pages 1225–1237. IEEE, 2015.

[108] S. Wang and J. Jiang. A compare-aggregate model for matching text sequences. *International Conference on Learning Representations*, 2017.

[109] Y. Wang, S. Feng, D. Wang, G. Yu, and Y. Zhang. Multi-label chinese microblog emotion classification via convolutional neural network. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, 2016.

[110] Y. Wang and A. Pal. Detecting emotions in social media: A constrained optimization approach. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[111] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Towards universal paraphrastic sentence embeddings. *International Conference on Learning Representations*, 2016.

[112] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[113] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008.

[114] R. Xia, X. Hu, J. Lu, J. Yang, and C. Zong. Instance selection and instance weighting for cross-domain sentiment classification via PU learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2013.

[115] R. Xia, F. Xu, J. Yu, Y. Qi, and E. Cambria. Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis. *Information Processing & Management*, 52(1):36–45, 2016.

[116] R. Xia, J. Yu, F. Xu, and S. Wang. Instance-based domain adaptation in NLP via in-target-domain logistic approximation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[117] M. Xiao and Y. Guo. Feature space independent semi-supervised domain adaptation via kernel matching. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):54–66, 2015.

[118] R. Yan, Y. Song, and H. Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR*, 2016.

[119] P. Yang, W. Gao, Q. Tan, and K.-F. Wong. Information-theoretic multi-view domain adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2012.

[120] Y. Yang and J. Eisenstein. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[121] Y. Yang and J. Eisenstein. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2015.

[122] Z. Yang, R. Salakhutdinov, and W. W. Cohen. Transfer learning for sequence tagging with hierarchical recurrent networks. *International Conference on Learning Representations*, 2017.

[123] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2016.

[124] W. Yin and H. Schütze. Convolutional neural network for paraphrase identification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2015.

[125] W. Yin, H. Schütze, B. Xiang, and B. Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

[126] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 2014.

[127] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[128] J. Yu and J. Jiang. 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)*, 2015.

[129] J. Yu and J. Jiang. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[130] J. Yu and J. Jiang. Pairwise relation classification with mirror instances and a combined convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016.

[131] J. Yu and J. Jiang. Learning auxiliary tasks for document-level cross-domain sentiment classification. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, 2017.

[132] J. Yu, L. Marujo, J. Jiang, P. Karuturi, and B. William. Improving multi-label emotion classification via sentiment classification with dual attention transfer network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[133] J. Yu, M. Qiu, J. Jiang, J. Huang, S. Song, W. Chu, and H. Chen. Modelling domain relationships for transfer learning on retrieval-based question answering systems in E-commerce. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[134] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 2015.

[135] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010.

[136] D. Zhou, Y. Yang, and H. Yulan. Relevant emotion ranking from text constrained with emotion relationships. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2018.

[137] D. Zhou, X. Zhang, Y. Zhou, Q. Zhao, and X. Geng. Emotion distribution learning from texts. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[138] G. Zhou, Z. Xie, J. X. Huang, and T. He. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[139] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu. Emotional chatting machine: emotional conversation generation with internal and external memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[140] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He. Supervised representation learning: Transfer learning with deep autoencoders. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.