Syntactically Meaningful and Transferable Recursive Neural Networks for Aspect and Opinion Extraction

Wenya Wang
Nanyang Technological University
Singapore
School of Computer Science and Engineering
wangwy@ntu.edu.sg

Sinno Jialin Pan
Nanyang Technological University
Singapore
School of Computer Science and Engineering
sinnopan@ntu.edu.sg

In fine-grained opinion mining, extracting aspect terms (a.k.a. opinion targets) and opinion terms (a.k.a. opinion expressions) from user-generated texts is the most fundamental task in order to generate structured opinion summarization. Existing studies have shown that the syntactic relations between aspect and opinion words play an important role for aspect and opinion terms extraction. However, most of the works either relied on predefined rules or separated relation mining with feature learning. Moreover, these works only focused on single-domain extraction, which failed to adapt well to other domains of interest where only unlabeled data are available. In real-world scenarios, annotated resources are extremely scarce for many domains, motivating knowledge transfer strategies from labeled source domain(s) to any unlabeled target domain. We observe that syntactic relations among target words to be extracted are not only crucial for single-domain extraction, but also serve as invariant "pivot" information to bridge the gap between different domains. In this article, we explore the constructions of recursive neural networks based on the dependency tree of each sentence for associating syntactic structure with feature learning. Furthermore, we construct transferable recursive neural networks to automatically learn the domain-invariant fine-grained interactions among aspect words and opinion words. The transferability is built on an auxiliary task and a conditional domain adversarial network to reduce domain distribution difference in the hidden spaces effectively in word level through syntactic relations. Specifically, the auxiliary task builds structural correspondences

Submission received: 9 December 2018; revised version received: 16 July 2019; accepted for publication: 17 September 2019.

https://doi.org/10.1162/COLI_a_00362

across domains by predicting the dependency relation for each path of the dependency tree in the recursive neural network. The conditional domain adversarial network helps to learn domain-invariant hidden representation for each word conditioned on the syntactic structure. In the end, we integrate the recursive neural network with a sequence labeling classifier on top that models contextual influence in the final predictions. Extensive experiments and analysis are conducted to demonstrate the effectiveness of the proposed model and each component on three benchmark data sets.

1. Introduction

Fine-grained opinion mining (Pang and Lee 2008; Liu 2011) aims to extract important information from opinionated texts. It consists of several subtasks, including aspect/opinion terms extraction (Hu and Liu 2004; Qui et al. 2011; Yin et al. 2016; Wang et al. 2017), aspect categorization (Lu, Zhai, and Sundaresan 2009; Zhao et al. 2010; Chen, Mukherjee, and Liu 2014; Lakkaraju, Socher, and Manning 2014; He et al. 2017), and aspect-dependent sentiment classification (Jiang et al. 2011; Dong et al. 2014; Ma et al. 2017). Among them, aspect and opinion terms extraction serves as the most fundamental task, which involves the identification of explicit entity features or attributes from user-generated texts, along with the opinions being expressed. For example, in a restaurant review "*They offer good appetizers*," the aspect term to be extracted is *appetizers*, and the opinion term is *good*.

Many existing studies have focused on single-domain aspect/opinion terms extraction, which can be classified into two categories: supervised learning and unsupervised learning. Under supervised learning, the task is formulated as a sequential labeling problem that predicts a label for each token. For example, Jin and Ho (2009), and Li et al. (2010) applied graphical models with extensive feature engineering for each token, including lexicon-based features and syntactic features. Deep learning models were also proposed to learn high-level features for each token (Liu, Joty, and Meng 2015; Yin et al. 2016; Li and Lam 2017; Wang et al. 2017). However, Liu, Joty, and Meng (2015) only used recurrent neural networks to model the sequential relations without considering syntactic structures. Yin et al. (2016) learned relation embeddings as a pretraining step that was separated from discriminative feature learning for final predictions. Wang et al. (2017) and Li and Lam (2017) applied attention models without explicit relation information. For unsupervised learning, most approaches are designed based on a set of predefined rules. Hu and Liu (2004) applied association rule mining to extract product features and Qiu et al. (2011) manually constructed some rules that associate aspect words and opinion words through syntactic relations. These rules are used to augment the set of target terms iteratively from a seed collection. However, these approaches highly rely on hand-coded rules and are inflexible. Existing work shows that syntactic structure plays an important role for propagating information between aspect and opinion words, yet they fail to associate feature learning with syntactic relation learning. To overcome the aforementioned limitation, we propose a dependency-tree-based recursive neural network that integrates syntactic tree structure into the deep model. In our preliminary work (Wang et al. 2016), we have developed this deep architecture for single-domain extraction and have demonstrated promising results. Based on this intuition and our observation that syntactic relations are invariant across different domains, we further develop a transferable recursive neural network to solve the more challenging problem of cross-domain extraction.

Note that existing supervised approaches cannot be directly applied for cross-domain extraction because a model trained on the source domain is not applicable for the target domain due to the large domain shift. Only a few unsupervised domain adaptation approaches have been proposed to transfer word-level knowledge from a labeled source domain to the unlabeled target domains for fine-grained extraction. Li et al. (2012) proposed a bootstrap method to iteratively expand the opinion and aspect lexicons in the target domain by exploiting source-domain labeled data and cross-domain common relations between the aspect terms and opinion terms. Ding, Yu, and Jiang (2017) proposed integrating auxiliary supervisions based on predefined rules on top of a recurrent neural network to learn domain-invariant hidden representation for each word. However, both methods require pre-mined syntactic patterns and the prior knowledge of a sentiment lexicon as a bridge. On the one hand, though the existing domain adaptation methods (Li et al. 2012; Ding, Yu, and Jiang 2017) have shown that syntactic relations or patterns among the aspect words and opinion words are invariant across different domains and are crucial as the pivot knowledge to bridge the gap, they fail in terms of flexibility and error propagation. On the other hand, as mentioned earlier, we proposed a dependency-tree-based recursive neural network for modeling syntactic interactions automatically through information propagation. Inspired by these two points, we further develop a transferable recursive neural network (TRNN) to transfer feature learning across domains through domain-invariant syntactic relations. Different from the single-domain model, TRNN encodes each dependency relation into a vector, on which an auxiliary task is integrated to predict its corresponding dependency relation category. The auxiliary task helps to cluster relation vectors with similar dependencies across domains, which builds structural correspondences across different domains. To explicitly align word feature spaces for source and target domains, we incorporate a conditional domain adversarial network (cDAN) that takes word features and their corresponding dependency relations as the input. Inspired by Ganin et al. (2016), the cDAN involves a domain discriminator that competes with the extraction model to align source and target spaces conditioned on syntactic structures. We observe that dependency relations may not be accurate when the parser is not perfect, which brings inevitable noise to the learning process. This can be alleviated through an auto-encoder that clusters the relations into their intrinsic relation groups. In the end, we further incorporate a sequence model, namely, a recurrent neural network, to exploit the contextual interactions within a sentence. Extensive analysis is provided to investigate the effectiveness of the proposed model for knowledge transfer.

Compared with our preliminary work (Wang et al. 2016; Wang and Pan 2018), the contributions of this article are summarized as follows.

- We integrate both of the preliminary works with the core idea on exploring recursive neural networks for both single-domain and cross-domain aspect/opinion terms extraction. We introduce how we extend the single-domain model toward the cross-domain setting.
- Besides constructing the auxiliary task as in Wang and Pan (2018), we further introduce a conditional domain adversarial network in the model to improve the knowledge transferability across domains.
- We conduct more extensive experiments to verify the effectiveness of the proposed model for cross-domain, fine-grained sentiment analysis.

2. Related Work

2.1 Aspect and Opinion Term Extraction

The problem of aspect/opinion terms extraction has been actively studied. Among the earliest work, Hu and Liu (2004) proposed using association rule mining to extract product aspects, and use synonyms and antonyms in WordNet to extract opinion terms through seed opinion set expansion. In follow-up work, syntactic relations among aspect words and opinion words were further exploited for the extraction task (Popescu and Etzioni 2005; Wu et al. 2009; Qiu et al. 2011). As a typical example, Qiu et al. (2011) adopted manually designed rules associating syntactic relations to double propagate and augment the sets of aspect words and opinion words. These rules usually deduce an aspect or opinion word if it belongs to some POS tags and has certain dependency relations with other extracted words. However, these methods all heavily depend on predefined rules for extraction and are restricted to specific types of POS tags for aspects and opinions, making them inflexible. As another category, topic models were also proposed for aspect extraction (Mei et al. 2007; Titov and McDonald 2008; Lu, Zhai, and Sundaresan 2009; Li, Huang, and Zhu 2010; Zhang et al. 2010; Chen, Mukherjee, and Liu 2014). For instance, Mei et al. (2007) used a mixture model based on both aspect and sentiment models. Titov and McDonald (2008) proposed a multigrain topic model that is able to discover both global topics and local topics. Li, Huang, and Zhu (2010) proposed the MaxEnt-LDA hybrid model to jointly discover both aspect and opinion words, which can leverage syntactic features. However, topic models are not suitable for explicit target terms extraction within each sentence and are difficult to align topics to their semantic meanings.

These models are referred to as unsupervised learning methods, which achieve inferior results compared with supervised methods that utilize label information for discriminative feature learning. For supervised learning, the task is modeled as a sequence labeling problem. In Jin and Ho (2009), Jakob and Gurevych (2010), Li et al. (2010), and Ma and Wan (2010) graphical models, for example, hidden Markov models or conditional random fields (CRFs), were proposed to learn context interactions among the input sentence according to the connections in a graph. These methods rely on rich hand-crafted features and do not consider interactions between aspect terms and opinion terms explicitly. Another direction applied a word alignment model to capture opinion relations among a sentence (Liu, Xu, and Zhao 2012; Liu et al. 2013). The methods in the direction, however, require sufficient data to identify desired relations. Recently, a number of deep learning models have been proposed (Liu, Joty, and Meng 2015; Zhang, Zhang, and Vo 2015; Yin et al. 2016; Li and Lam 2017; Wang et al. 2017; Xu et al. 2018). Liu, Joty, and Meng (2015) applied a recurrent neural network on top of pretrained word embeddings for aspect extraction. Yin et al. (2016) proposed learning dependency path embeddings to explore relations among words, but failed to incorporate relation embedding learning into a joint model that may propagate label errors to feature learning. Wang et al. (2016) proposed using the dependency-tree-based recursive neural network to learn high-level aspect-opinion interactions. Wang et al. (2017) and Li and Lam (2017) proposed using an attention mechanism for extracting target words without explicit syntactic relations. When it comes to cross-domain extraction, existing supervised methods usually fail, because the aspect terms from two different domains are usually disjoint (e.g., *laptop* vs. *restaurant*), leading to a large domain shift in the feature vector space.

2.2 Domain Adaptation

Many domain adaptation methods have been proposed for cross-domain sentence-level or document-level sentiment classification. Some of them aim to utilize pivot information that are shared across domains as a bridge to align different feature spaces (Blitzer, Dredze, and Pereira 2007; Pan et al. 2010; Bollegala, Maehara, and ichi Kawarabayashi 2015; Yu and Jiang 2016). Another group of work directly learn shared spaces across different domains through projection via auto-encoders (Glorot, Bordes, and Bengio 2011; Chen et al. 2012; Zhou et al. 2016) or the domain adversarial network (Li et al. 2017). Few studies have addressed the problem of cross-domain fine-grained opinion extraction. Jakob and Gurevych (2010) proposed a cross-domain CRF that is built on nonlexical invariant feature engineering (e.g., POS tags and dependency relations). Li et al. (2012) developed a boosting-style method to iteratively expand the aspect and opinion sets in the target domain through manually defined rules incorporating common syntactic relations among aspect and opinion words across domains. Ding, Yu, and Jiang (2017) explored pivot knowledge on opinion lexicon and syntactic relations to build correspondences across domains and applied a recurrent neural network with auxiliary tasks to learn domain-invariant word representations. However, existing methods largely depend on predefined pivot information that are fixed and inflexible. Recently, we proposed a recursive neural structural correspondence network (Wang and Pan 2018) that integrates an auxiliary task on dependency relation prediction to build structural correspondences across domains. In this work, we further extend our previous idea for more accurate cross-domain aspect and opinion terms extraction.

3. Problem Definition and Motivation

3.1 Problem Definition

Formally, the task of aspect and opinion terms extraction can be formulated as a sequence tagging problem. The input is a sequence of words $s = \{w_1, \ldots, w_n\}$ with pretrained word embeddings $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}_i \in \mathbb{R}^D$ is the word embedding for w_i. The objective is to predict a label for each word, resulting in a label sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$, with $y_i \in \mathbf{L}$. We adopt the BIO encoding scheme with five different classes, namely $\mathrm{L} = \{\mathrm{BA}, \mathrm{IA}, \mathrm{BO}, \mathrm{IO}, \mathrm{N}\}$, where BA or BO represents the beginning of aspect term or opinion term, respectively. IA or IO represents the inside

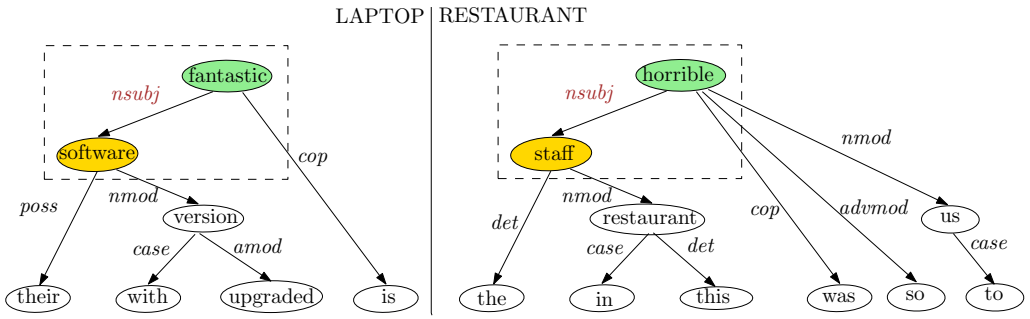


Figure 1
An example of two reviews in different domains with similar syntactic patterns.

position of aspect term or opinion term, respectively, and N represents "None of the above." Note that a sequence of predictions with BA at the beginning followed by IA indicates a multi-word aspect term, which is similar for opinion terms. For cross-domain aspect and opinion terms extraction, we are given a set of labeled data in the source domain $\mathcal{D}_S = \{(\mathbf{x}_i^S, \mathbf{y}_i^S)\}_{i=1}^{N_S}$ as well as a set of unlabeled data in the target domain $\mathcal{D}_T = \{(\mathbf{x}_j^T)\}_{j=1}^{N_T}$. The objective is to transfer knowledge from source domain to the target domain in order to obtain label sequences $\mathbf{y}^T$ in the target domain.

For ease of illustration, we first introduce some definitions. A syntactic structure for each sentence is reflected via a dependency tree as shown in Figure 1, which can be obtained from a dependency parser. A dependency path, for example, *horrible* $\xrightarrow{\text{nsubj}}$ *staff*, consists of a child word *staff*, a parent word *horrible*, and a dependency relation (DR) "nsubj." We name this relation as upward DR for its child node and downward DR for its parent node.

3.2 Motivation

Syntactic structure has been shown to be crucial for aspect and opinion terms extraction because there are certain dependency relations existing among the aspect and opinion words for information propagation. For example, given the sentence "the staff in this restaurant was so horrible to us," the aspect term *staff* and the opinion term *horrible* has the dependency relation nsubj. Given that *horrible* is already extracted and the relation nsubj is frequently observed between aspect and opinion words, the aspect word *staff* should be identified accordingly. Most existing work either encodes the relations into manual-designed rules or focuses on learning semantically meaningful word features through deep neural networks without associating syntactic information. We propose a remedy to this problem by incorporating syntactic structures into the process of feature learning. We name the proposed model a single-domain recursive neural network (SRNN), which is constructed according to the dependency tree of each sentence. An illustration of the recursive process is shown on the left-hand side of Figure 2. The model computes the hidden representation for each parent node in the tree hierarchy based on its own embedding as well as the hidden representations of its child nodes. As a result, the hidden space encodes the features from their dependents according to the syntactic structure, thus propagating information among syntactically related words.

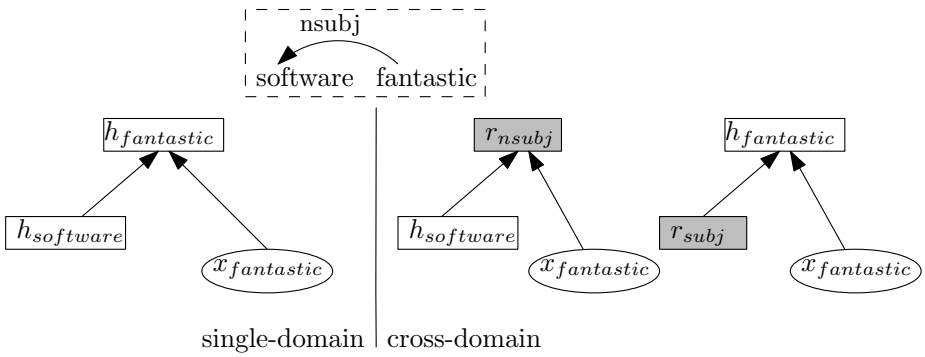


Figure 2
An illustration of different recursive structures.

Different from sentence- or document-level knowledge transfer, aspect and opinion term extraction across domains require fine-grained word-level adaptation, which is much more challenging. Existing work has designed hand-coded rules and a sentiment lexicon as pivot information—for example, in Figure 1, given a review sentence "*their software with upgraded version is fantastic*" in the source domain and "*the staff in this restaurant was so horrible to us*" in the target domain. If *horrible* has been extracted as a common sentiment word, and "ASPECT-nsubj-OPINION" has been identified as a common syntactic pattern from the source domain, *staff* could be deduced as an aspect term using the identified syntactic pattern (Li et al. 2012). With a similar idea, Ding, Yu, and Jiang (2017) encoded these common patterns into specific rules that are integrated as auxiliary tasks into a recurrent neural network. These works indicate the importance of invariant syntactic information between words as a bridge between different domains. However, their drawback is reflected by restraining the pivot information within a few manually defined rules that are fixed and inflexible. Although the proposed SRNN is able to propagate information among aspect and opinion words automatically through the tree structure, this kind of interaction is hard to capture when label information is not available in the target domain. A crucial question to ask is how to transfer these important interactions from the source to the target domain.

To answer that, we propose a transferable recursive neural network with the following two characteristics: (1) Instead of hand-coded pivot knowledge, our model automatically learns the syntactic features through dependency-tree-based recursive neural networks. (2) The syntactic structure is treated as invariant features across different domains. We observe that the dependency relations are domain-invariant. When two pairs of words in the source and the target domains have the same dependency relation, the corresponding words tend to have similar functionalities. For example, *staff* and *horrible* in the restaurant domain is connected via the dependency relation nsubj, which also connects *software* and *fantastic* in the laptop domain. We shall observe that both *staff* and *software* are targeted subjects, and both *horrible* and *fantastic* are adjective qualifiers, irrespective of their domains. With this intuition, word-level knowledge across domains could be transferred via common syntactic structures. We propose two transferable modules based on dependency trees to realize knowledge transfer. The first module is an auxiliary task for predicting dependency relations between each connected word pair. The second module is a conditional domain adversarial network for projecting the representations of syntactically-similar words into a common space.

Specifically, TRNN is built on a recursive structure shown in the right-hand side of Figure 2. Different from SRNN, we also embed each DR into the distributed vector space. The model maps each DR into a distributed vector $\mathbf{r}_{\text{nsubj}}$, which can be regarded as a relation vector. The relation vector is then taken as the input to generate the hidden representation for the parent word. Based on this construction, the auxiliary module applies a relation classifier on top of the relation vector $\mathbf{r}_{\text{nsubj}}$ to predict which dependency relation it belongs to. This classifier is trained in a supervised manner with the ground-truth relation labels for both domains. At the same time, it draws those relation vectors in the same class close to each other. As a concrete example, in Figure 1, the relation vectors for the dependency paths "*software*-nsubj-*fantastic*" and "*staff*-nsubj-*horrible*" in both domains should lie in close proximity. As a result, *fantastic* and *horrible* are clustered in the hidden space. In a word, the auxiliary module transfers knowledge of word representations with similar syntactic functionalities across domains via dependency relations.

According to Figure 2, the auxiliary module only aligns the hidden representations for the words with the same downward DR, but ignores the situation when they have

the same upward DR. Moreover, the transferability implicitly depends on the learned relation vectors. To explicitly learn a domain-invariant hidden representation for each word and exploit the dependency correspondences for the other direction, we integrate a second transferable module: a cDAN that takes the concatenation of the hidden vector and the upward DR for each word as the input. Motivated by Ganin et al. (2016), a DAN consists of a domain discriminator that competes with the target model until it cannot differentiate the features between different domains. This enforces the model to learn domain-invariant representations for final predictions. The cDAN is introduced in Mirza and Osindero (2014) and Long et al. (2018) and incorporates class constraints when discriminating between different domains. In our problem setting, the domain discriminator is applied on top of the hidden vector space and is conditioned on the upward dependency relation for each word. However, the dependency relations are not guaranteed to be accurate, especially for user-generated texts. It might harm the learning process for both the auxiliary task and cDAN when some noise exists for certain relations. This problem of noisy labels has been addressed using perceptual consistency (Reed et al. 2015). Inspired by the taxonomy of dependency relations (de Marneffe and Manning 2008), dependencies with similar functionalities could be grouped together (e.g., *dobj*, *iobj*, and *pobj* all indicate objects). We propose using an auto-encoder to automatically group these relations in an unsupervised manner. The reconstruction loss serves as the consistency objective that reduces label noise by aligning relation features with their intrinsic relation group.

4. Preliminary Model

4.1 Domain Adversarial Network

The domain adversarial network was proposed in Ganin et al. (2016) for the problem of unsupervised domain adaptation. Unlike most traditional transfer learning approaches, which learn a mapping of features either from one domain to another domain or from both domains to a common space, DAN incorporates domain-distribution learning into the feature learning itself. The objective of learning a DAN is twofold: (1) The model should be discriminative for the main task in the source domain. (2) The model is indiscriminative for data distributions in different domains. In this case, DAN embeds the knowledge transfer process into the learning of representations, such that the final predictions can be obtained from the source classifier because the features are already invariant across domains. To achieve this objective, the DAN is designed in the following procedure.

Assume we have a multi-layer neural network $F(\cdot;\theta_f)$ as the feature extractor, where θ_f denotes the parameters involved in the neural network. Let $Y(\cdot;\theta_y)$ denote the final classification layer with parameters θ_y. Given an input instance $\mathbf{x}$, a typical forward computation usually generates $Y(F(\mathbf{x};\theta_f);\theta_y)$ as the final prediction that can be compared with the ground-truth labels for back-propagation. When two different domains are involved with labels provided only for the source domain, the feature extractor $F(\cdot;\theta_f)$ learned in the source domain is not applicable for the target domain due to the domain shift. To address this issue, a domain discriminator $D(\cdot;\theta_d)$ is incorporated in the learning process. Specifically, when feeding an instance $\mathbf{x}$ either from the source domain or target domain, the model will generate the following two outputs:

$$\mathbf{y} = Y(F(\mathbf{x};\theta_f);\theta_y)$$
$$\mathbf{d} = D(F(\mathbf{x};\theta_f);\theta_d)$$

with $\mathbf{y}$ and $\mathbf{d}$ indicating the main prediction and domain prediction, respectively. $\mathbf{d}$ is a two-dimensional vector, where each entry represents the probability that $\mathbf{x}$ comes from the source domain ($\mathbf{x} \sim \mathcal{D}_S$) or target domain ($\mathbf{x} \sim \mathcal{D}_T$). During training, we aim to minimize the domain discriminator loss, but at the same time maximize the domain confusion for the main predictor:

$$\min \frac{1}{n_S} \sum_{i=1}^{n_S} \mathcal{L}_y(Y(F(\mathbf{x}_i; \theta_f); \theta_y), \hat{\mathbf{y}}_i) - \frac{\lambda}{n_S + n_T} \sum_{i=1}^{n_S+n_T} \mathcal{L}_d(D(F(\mathbf{x}_i; \theta_f); \theta_d), \hat{\mathbf{d}}_i)$$

$$\min \frac{\lambda}{n_S + n_T} \sum_{i=1}^{n_S+n_T} \mathcal{L}_d(D(F(\mathbf{x}_i; \theta_f); \theta_d)\hat{\mathbf{d}}_i)$$

where $\mathcal{L}_y$ and $\mathcal{L}_d$ denote the loss for the main prediction task with true label $\hat{\mathbf{y}}_i$ and loss for the domain prediction task with true label $\hat{\mathbf{d}}_i$, respectively. The update process involves a gradient reversal layer for the domain predictor that results in the following updates:

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial \mathcal{L}_y}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d}{\partial \theta_f} \right)$$

$$\theta_y \leftarrow \theta_y - \mu \left(\frac{\partial \mathcal{L}_y}{\partial \theta_y} \right)$$

$$\theta_d \leftarrow \theta_d - \mu \left(\lambda \frac{\partial \mathcal{L}_d}{\partial \theta_d} \right)$$

The DAN as defined here assumes similar data distributions between the source and target domains in general. However, in many cases, the input data do not commit a universal but multimodal distributions. For example, when multiple classes exist, the data under each class can be regarded as following one unique distribution. Motivated by this point, Long et al. (2018) proposed a conditional domain adversarial network. Specifically, the domain discriminator is applied on the joint distribution of data and classes, instead of only on the input data. The addition of conditional information for the domain classifier can be revealed by replacing the original $D(F(\mathbf{x}; \theta_f); \theta_d)$ with $D(F(\mathbf{x}; \theta_f), \mathbf{g}; \theta_d)$, where $\mathbf{g}$ represents the class information. With this addition, the network is trained to align the source domain and the target domain conditioned on their corresponding classes.

5. Model Architecture

In this section, we begin with the description of SRNN for the single-domain extraction problem followed by TRNN for the cross-domain setting. Both of the proposed models are constructed from the dependency tree of each sentence and process in a recursive manner where higher-level nodes in the tree hierarchy are obtained after the computation of their child nodes. SRNN treats each different dependency relation as a transformation matrix that is involved in the forward computation for each node, whereas TRNN embeds each dependency path into a distributed vector that acts as a bridge to transfer knowledge across different domains. For final predictions, both single-domain and cross-domain RNNs are combined with a sequence modeling module gated recurrent unit (GRU), a special form of recurrent neural network.

The joint model takes the output of SRNN or TRNN as the input feature representations for GRU, and generates the final output for each word for classification. The detailed constructions are shown in the following sections.

5.1 Single-Domain Recursive Neural Networks

In our preliminary work (Wang et al. 2016), we proposed a dependency-tree-based recursive neural network for single-domain aspect and opinion terms extraction. Different from existing work, the proposed model incorporates the syntactic structure of a sentence into an automatic representation learning architecture, such that the important relation information could be propagated to the feature learning process. An example of the model structure is shown in Figure 3. Specifically, the network takes word embedding $\mathbf{x}_n$ for each word as the input and produces the hidden vector $\mathbf{h}_n$ as the output. The hidden vector is computed in a recursive manner according to the dependency tree of each sentence generated from a dependency parser. In Figure 3, the bottom arrows indicate the dependency relations for certain word pairs with the relation type shown under the arrows. Each relation connects a child word (end of the arrow) to a parent word and each word only has one parent. Given this dependency structure, we can build a hierarchical tree such that each node in the tree represents a word in the sentence. The parent word for each relation lies in higher hierarchy than its children.

The computation of the recursive neural network starts with the leaf nodes and recursively reaches to the root. Formally speaking, the hidden representation $\mathbf{h}_n$ for the nth word is computed as

$$\mathbf{h}_n = f(\mathbf{W}_v \cdot \mathbf{x}_n + \mathbf{b} + \sum_{k \in \mathcal{K}_n} \mathbf{W}_{r_{nk}} \cdot \mathbf{h}_k)$$

where $\mathbf{W}_v \in \mathbb{R}^{d \times d}$ is a transformation matrix for word embeddings. $\mathbf{W}_{r_{nk}} \in \mathbb{R}^{d \times d}$ is one of the relational transformation matrices for hidden vectors. We assign $|R|$ different relational transformation matrices, where $R = \{\text{nsubj}, \text{dobj}, \ldots\}$ is the set of different dependency relations. Hence $r_{nk} \in R$ indicates the dependency relation between the nth node and kth node. $\mathcal{K}_n$ is the set of child nodes that depend on node n. In a

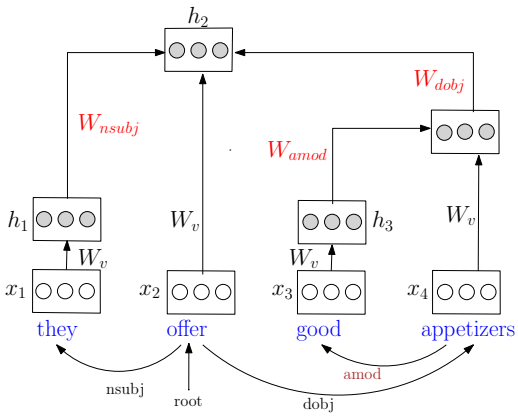


Figure 3
The structure of a dependency-tree-based recursive neural network.

word, the hidden vector for each node is obtained from its own word embedding, together with the hidden representations from its children. This construction embeds syntactic interactions among the input words into their hidden representations. To be more specific, consider the example shown in Figure 3. The dependency between the aspect word *appetizer* and the opinion word *good* is $\mathbf{x}_4$ (appetizers) $\xrightarrow{\text{amod}}$ $\mathbf{x}_3$ (good), which leads to the following equation

$$\mathbf{h_4} = f(\mathbf{W}_v \cdot \mathbf{x}_4 + \mathbf{b} + \mathbf{W}_{\text{amod}} \cdot \mathbf{h_3}) \tag{1}$$

We can observe that the feature for *good*, together with the relational transformation matrix, affects the final feature for *appetizers*. When this interaction between aspect words and opinion words is frequently seen in the training corpus, the model could learn this pattern to help predictions in the test phase.

However, this architecture is not directly applicable for the cross-domain setting. Given labeled source domain and unlabeled target domain, a straightforward idea is to share the parameters learned in the source domain for target predictions. Although sharing relational transformation matrices aligns with the fact that dependency relations are domain-invariant, this cannot guarantee learning of domain-invariant hidden representations. Specifically, when the two aspect words from two domains have disjoint features and children according to the dependency trees, the resultant hidden vectors are also dissimilar. This motivates a novel construction to facilitate knowledge transfer discussed in the following section.

5.2 Transferable Recursive Neural Networks

The overall architecture of TRNN is shown in Figure 4. Given an input sentence with pre trained word embedding $\mathbf{x}$ for each word, the model recursively computes the hidden representation $\mathbf{h}$ for each word as the output of TRNN. Meanwhile, knowledge transfer takes place via two transferable components: an auxiliary task as well as a conditional domain adversarial network are adopted to learn domain-invariant representations. Specifically, given a dependency path, TRNN first produces a relation vector $\mathbf{r}_{ij}$ through its connected words. Then an auxiliary task takes $\mathbf{r}_{ij}$ as the input for classifying its dependency relation (nsubj in this case). To reduce the negative effect brought by noisy relation labels generated from imperfect parsers, an auto-encoder is incorporated to generate relation clusters $\mathbf{y}_{ij}^G$ as an encoding step. The decoding process produces exact relation label $\mathbf{y}_{ij}^R$ as well as reconstructing the input relation vector. The learned relation vectors serve as a bridge across different domains when they are used to compute $\mathbf{h}_j$ as the hidden representation for its parent word. To explicitly map two domains into a common space, a cDAN is adopted with a domain discriminator. The input to the discriminator is a concatenation of $\mathbf{h}_i$ and one-hot relation feature $\mathbf{y}_{ij}^G$ representing the relation cluster to which the upward dependency relation belongs. This construction aims to align word features across different domains based on their syntactic structure. We describe each component in detail in the following sections.

5.2.1 Recursive Neural Networks with Auxiliary Tasks. Similar to SRNN, TRNN is recursively constructed according to the pregenerated dependency tree for each sentence. To make the model transferable across different domains, we embed each dependency relation in the tree into a distributed vector space, which can be taken as the input to an auxiliary task. For concrete illustration, we present the RNN with auxiliary tasks

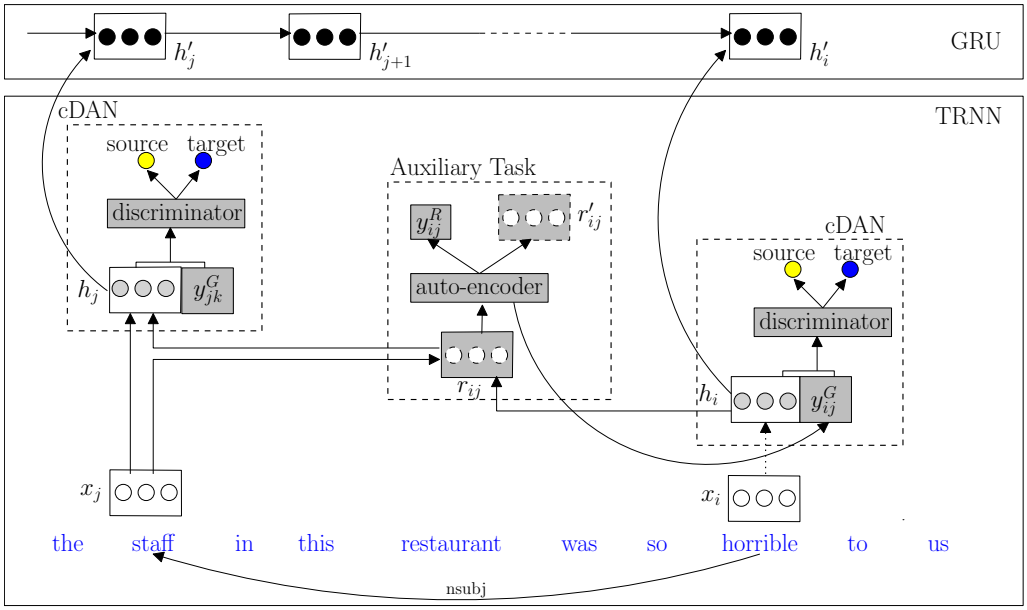


Figure 4
An overview of the proposed model.

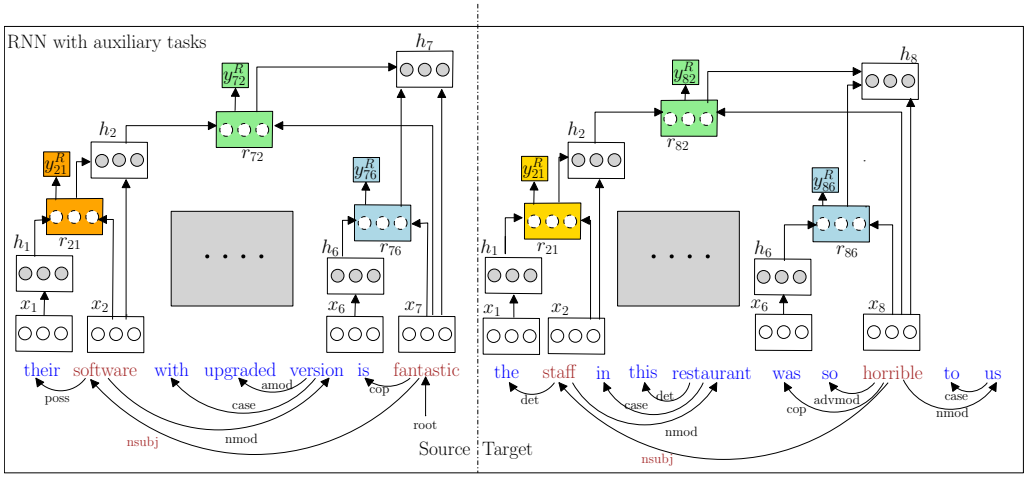


Figure 5
An example of RNN with auxiliary tasks for both source and target domains.

for both source and target domain in Figure 5. The model computes the hidden representation $\mathbf{h}_i$ for each word recursively from leaf nodes to higher-level nodes in the tree. Instead of using child node features as in SRNN, we incorporate a dependency relation vector to produce the hidden representation for the parent nodes. Specifically, we begin with input word embedding $\mathbf{x}_i \in \mathbb{R}^D$ for each node. Consider the source-domain sentence shown in Figure 5 as an example. For ease of illustration, we ignore some words in the middle of the sentence. The hidden vectors for the leaf nodes are first

generated through a non-linear transformation on the input vectors:

$$\mathbf{h}_1 = \tanh(\mathbf{W}_x\mathbf{x}_1 + \mathbf{b}) \text{ and } \mathbf{h}_6 = \tanh(\mathbf{W}_x\mathbf{x}_6 + \mathbf{b})$$

with $\mathbf{W}_x \in \mathbb{R}^{d\times D}$ as a transformation matrix. To obtain hidden vectors for non-leaf nodes, a relation vector $\mathbf{r}_{ij} \in \mathbb{R}^d$ is generated first for each dependency path that connects them to their child node. Here, i and j denote the indices of the parent and child word of the dependency edge, respectively. For example, the relation vector for $\mathbf{x}_7$ (fantastic) $\xrightarrow{\text{nsubj}}$ $\mathbf{x}_2$ (software) is computed as

$$\mathbf{r}_{72} = \tanh(\mathbf{W}_h\mathbf{h}_2 + \mathbf{W}_x\mathbf{x}_7)$$

Then the hidden vector for *fantastic* can be obtained through

$$\mathbf{h}_7 = \tanh(\mathbf{W}_{\text{nsubj}}\mathbf{r}_{72} + \mathbf{W}_x\mathbf{x}_7 + \mathbf{b})$$

by the transformation of its own input embedding as well as the relation path vectors that connect it to its dependents. Then an auxiliary task is integrated on relation vectors to predict its corresponding dependency relation type:

$$\hat{\mathbf{y}}^R_{72} = \text{softmax}(\mathbf{W}_R\mathbf{r}_{72} + \mathbf{b}_R)$$

where $\mathbf{W}_R \in \mathbb{R}^{K\times d}$ is the relation classification matrix. The auxiliary task applies to each dependency path and is a K-class supervised classification problem with K representing the total number of different dependency relations in the corpus. During training, the supervision of relation labels $\mathbf{y}^R_{ij} \in \mathbb{R}^K$ can be obtained for both source and target domains via the dependency parser. Through this auxiliary task, the relation vectors $\{\mathbf{r}_{ij}\}$'s are grouped together according to relation types, that is, similar or the same dependency relations from the source and target domains are projected within close proximity in the vector space. As the example shown in Figure 5, $\mathbf{r}_{72}$ for *fantastic* $\xrightarrow{\text{nsubj}}$ *software* and $\mathbf{r}_{82}$ for *horrible* $\xrightarrow{\text{nsubj}}$ *staff* are close to each other. As a result, these relation vectors could be regarded as a bridge for word knowledge transfer across domains: The hidden representations of the parent words ($\mathbf{h}_7$ for *fantastic* and $\mathbf{h}_8$ for *horrible*) for these relations are aligned, irrespective of the domains.

In general, the hidden representation $\mathbf{h}_i$ for the i-th node is produced through

$$\mathbf{h}_i = \tanh\Big(\sum_{j\in\mathcal{M}_i}\mathbf{W}_{R_{ij}}\mathbf{r}_{ij} + \mathbf{W}_x\mathbf{x}_i + \mathbf{b}\Big) \quad (2)$$

$$\text{where } \mathbf{r}_{ij} = \tanh(\mathbf{W}_h\mathbf{h}_j + \mathbf{W}_x\mathbf{x}_i)$$

Here $\mathcal{M}_i$ is the set of child nodes of w_i, and $\mathbf{W}_{R_{ij}}$ is the relation transformation matrix tied with each relation R_{ij}. For leaf nodes, the first term on the right-hand side of Equation (2) is removed. The predicted relation label vector $\hat{\mathbf{y}}^R_{ij}$ for $\mathbf{r}_{ij}$ is

$$\hat{\mathbf{y}}^R_{ij} = \text{softmax}\,(\mathbf{W}_R \cdot \mathbf{r}_{ij} + \mathbf{b}_R) \quad (3)$$

We adopt the cross-entropy loss for relation classification between the predicted label vector $\hat{\mathbf{y}}_{ij}^{R}$ and the ground-truth $\mathbf{y}_{ij}^{R}$ to encode relation side information into feature learning:

$$\ell_R = \sum_{n=1}^{N} -\mathbf{y}_{ij[n]}^{R} \log \hat{\mathbf{y}}_{ij[n]}^{R} \tag{4}$$

Through the auxiliary task, similar dependency relations in source and target domains are trained to be clustered in the relation vector space, which are then used to build correspondences between their parent words across two domains via Equation (2). In this case, words with similar syntactic functionalities are projected within close distance for both domains. Hence, the final word classifier trained from the source domain can be directly transferred to the target domain.

5.2.2 Addition with Conditional Domain Adversarial Networks. The auxiliary task introduced in Section 5.2.1 implicitly maps the hidden representations across different domains into a shared space through clustered relation vectors. To achieve more explicit adaptations, we further apply a cDAN with a domain discriminator to learn disentangled and transferable representations for domain adaptation. However, directly applying a cDAN for each word only assumes unitary distributions for both source and target domains. In fact, multimodal distributions should be observed for input words corresponding to different syntactic functionalities. To address this point, and motivated by Long et al. (2018), we adopt conditional cDANs that condition the domain discriminator on the dependency structure of a sentence. Specifically, the domain discriminator $\mathcal{D}$ generates a probability distribution $P(\mathcal{D}|\mathbf{h}, \mathbf{y}^{G})$ as already shown in Figure 4, where $\mathbf{y}^{G}$ refers to the relation label feature. We denote $\mathcal{D} = 1$ as the source domain and $\mathcal{D} = 0$ as the target domain. Then the domain discriminator computes

$$\hat{\mathbf{y}}_i^{d} = P(\mathcal{D}|\mathbf{h}_i, \mathbf{y}_{mi}^{G}) = \text{softmax}\left(\mathbf{W}_d[\mathbf{h}_i : \mathbf{y}_{mi}^{G}] + \mathbf{b}_d\right) \tag{5}$$

where m indicates the index of the parent node for w_i. Note that the dependency tree assigns each node with only one parent, hence m is unique. [:] represents concatenation of vectors. $\mathbf{y}_{mi}^{G}$ is a one-hot vector with entry 1 representing the index of the corresponding dependency relation between w_i and w_m. The domain discriminator is learned by minimizing the classification error of distinguishing the source from the target domains. At the same time, the main prediction model learns transferable representations that are indistinguishable by the domain discriminator. This process corresponds to the following two objectives:

$$\min \sum_{i=1}^{n_S} \ell_y(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \gamma \sum_{i=1}^{n_S+n_T} \ell_d(\mathbf{y}_i^{d}, \hat{\mathbf{y}}_i^{d}) \tag{6}$$

$$\min \gamma \sum_{i=1}^{n_S+n_T} \ell_d(\mathbf{y}_i^{d}, \hat{\mathbf{y}}_i^{d}) \tag{7}$$

where ℓ_y is the cross-entropy loss between final ground-truth labels and word-level predictions. Similarly, ℓ_d is the cross-entropy loss for domain classifier. γ controls the tradeoff between the main prediciton loss and the domain confusion loss. We apply the

cDAN on each node in the tree. On the one hand, by confusing the domain discriminator on the concatenation of hidden vectors and relation features from different domains, the model is able to align source and target domains on the joint distributions of word features and dependency structures. On the other hand, the conditioned information refers to the upward dependency relation for each word. Compared with the auxiliary task, which focuses on learning transferable word features based on the downward dependency relations, cDAN works in the other direction and further improves the transferability of the model when combining with the auxiliary module. That is, these two transferable modules aim at learning syntactically sensitive word representations that are transferable across domains.

5.2.3 Reduce Label Noise with Auto-encoders. The dependency relations are crucial for transferring knowledge across different domains. As discussed in the previous sections, the relations serve as auxiliary labels as well as conditional features for cDAN, hence, the accuracy of generated dependencies substantially affects the final prediction performance. However, the dependency parsers are not perfect and could produce incorrect dependency relations. Moreover, if we treat each unique relation as one class, there will be more than 40 different classes for the auxiliary task, which makes the classifier hard to learn. To resolve this problem, we propose integrating an auto-encoder into TRNN to cluster dependency relations and reduce the dimensions. We assume that there is a set of latent groups of relations: $G = \{1, 2, \ldots, |G|\}$, where each relation belongs to only one group. For the auxiliary task of relation predictions, an auto-encoder is applied on top of the relation vector $\mathbf{r}_{ij}$ before feeding it into the auxiliary classifier (Equation (3)). The goal is to encode the relation vector to a probability distribution of assigning this relation to any group. As can be seen Figure 6, each relation vector $\mathbf{r}_{ij}$ is first passed through the auto-encoder as follows:

$$p(G_{ij} = k|\mathbf{r}_{ij}) = \frac{\exp(\mathbf{r}_{ij}^{\top}\mathbf{W}_{enc}\mathbf{g}_k)}{\sum_{k' \in G} \exp(\mathbf{r}_{ij}^{\top}\mathbf{W}_{enc}\mathbf{g}'_k)} \tag{8}$$

where G_{ij} denotes the inherent relation group for $\mathbf{r}_{ij}$, $\mathbf{g}_k \in \mathbb{R}^d$ represents the feature embedding for group k, which is randomly initialized. $\mathbf{W}_{enc} \in \mathbb{R}^{d \times d}$ is the encoding matrix that computes bilinear interactions between relation vector $\mathbf{r}_{ij}$ and relation group

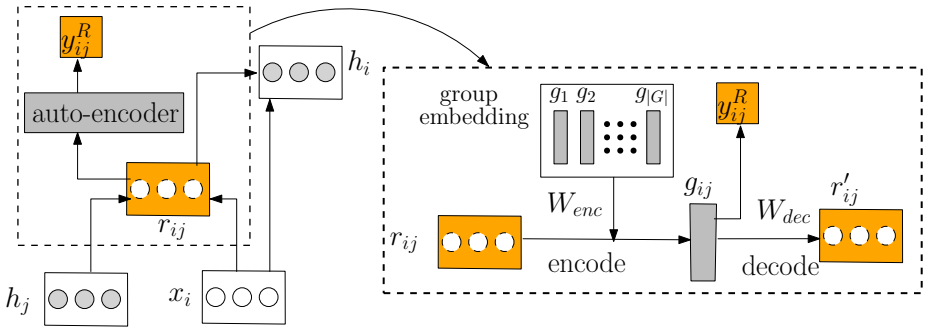


Figure 6
An auto-encoder for relation grouping.

embedding $\mathbf{g}_k$. Thus, $p(G_{ij} = k|\mathbf{r}_{ij})$ represents the probability of $\mathbf{r}_{ij}$ being mapped to group k. Different from traditional auto-encoders, where the encoding process generates a continuous hidden vector, the auto-encoder used in this model has a discrete form where the encoding process generates a probability distribution for group assignment as shown in Equation (8). Directly conducting decoding from $p(G_{ij} = k|\mathbf{r}_{ij})$ is meaningless, because the semantic meanings are not aligned between probability distribution and the input relation vector. To address this issue, we compute an accumulated relation group embedding for $\mathbf{r}_{ij}$ as

$$\mathbf{g}_{ij} = \sum_{k=1}^{|G|} p(G_{ij} = k|\mathbf{r}_{ij})\mathbf{g}_k \tag{9}$$

For decoding, the decoder takes $\mathbf{g}_{ij}$ as input and tries to reconstruct the relation feature input $\mathbf{r}_{ij}$. Moreover, $\mathbf{g}_{ij}$ is also used as the higher-level feature vector for $\mathbf{r}_{ij}$ for predicting the relation label. Therefore, the objective for the auxiliary task in Equation (4) becomes:

$$\ell_R = \ell_{R_1} + \alpha\ell_{R_2} + \beta\ell_{R_3} \tag{10}$$

where

$$\ell_{R_1} = \left\|\mathbf{r}_{ij} - \mathbf{W}_{dec}\mathbf{g}_{ij}\right\|_2^2 \tag{11}$$

$$\ell_{R_2} = \sum_{n=1}^{N} -\mathbf{y}^R_{ij[n]} \log \hat{\mathbf{y}}^R_{ij[n]} \tag{12}$$

$$\ell_{R_3} = \left\|\mathbf{I} - \bar{\mathbf{G}}^\top \bar{\mathbf{G}}\right\|_F^2 \tag{13}$$

Here ℓ_{R_1} is the reconstruction loss with $\mathbf{W}_{dec}$ being the decoding matrix, ℓ_{R_2} follows Equation (4) with $\hat{\mathbf{y}}^R_{ij} = \text{softmax}\ (\mathbf{W}_R\mathbf{g}_{ij} + \mathbf{b}_R)$, and ℓ_{R_3} is the regularization term on the correlations among latent groups with $\mathbf{I}$ being the identity matrix and $\bar{\mathbf{G}}$ being a normalized group embedding matrix that consists of normalized $\mathbf{g}_k$s as column vectors. This regularization term enforces orthogonality between $\mathbf{g}_k$ and $\mathbf{g}_{k'}$ for $k \neq k'$. Indeed, we expect the auto-encoder to learn meaningful relation groups such that intra-group relations are close to each other, whereas inter-group relations are distinct from each other (achieved by an orthogonality constraint on relation groups). We will qualitatively show later in the experiments that similar dependency relations are clustered after training, for example, iobj and dobj belong to the same group. α and β are used to control the trade-off among different losses. With the auto-encoder, the auxiliary task of relation classification is conditioned on group assignment. The reconstruction loss further ensures the consistency between relation features and groupings, which is supposed to dominate classification loss when the observed labels are inaccurate.

To reduce the negative effect of noisy relations on cDAN, we replace the explicit one-hot relation feature with relation cluster labels. Specifically, $\mathbf{y}^G_{mi}$ in Equation (5) becomes a one-hot vector with $\mathbf{y}^G_{mi}[k] = 1$, where k indicates the index of the relation group it belongs to. We can treat k as a pseudo-label for relation clusters, which is obtained as $k = \arg\max p(G_{mi}|\mathbf{r}_{mi})$ given Equation (8) computed from an auto-encoder.

5.3 Joint Training with Sequence Prediction Models

The recursive neural networks mainly capture syntactic interactions among a sequence, but ignore sequential correlations. To address this limitation, we propose a joint model that consists of two components: The first component is a SRNN or TRNN for single-domain or cross-domain setting, respectively, to explore syntactic interactions among aspect and opinion words. The second component is a GRU, which is a variant of recurrent neural networks to model contextual interactions among a sentence. GRU is able to learn long-term dependencies through gating units compared to standard recurrent neural networks, and at the same time less prone to over-fitting compared to long short-term memory (LSTM). Unlike our preliminary work (Wang et al. 2016), which adopts CRF as the sequential model, we choose GRU because of its ability to learn high-level feature interactions and its smoother integration with recursive neural networks. The resultant joint model combines both syntactic and sequential influences that are both crucial for the extraction task. Here we use SRNN-GRU/TRNN-GRU to denote the final joint model.

Besides syntactic structures, sequential interactions are also crucial for aspect and opinion terms extraction, for example, a token labeled as BA could infer the following token with high probability of being IA in a multi-word aspect term. Given label N for the current token, it is impossible for the next token to have label IA or IO. To incorporate such sequential information, we propose integrating a GRU-based recurrent neural network on top of RNN to form a joint model. In this case, the input for GRU is the hidden representations $\mathbf{h}_i$ learned by SRNN/TRNN for the i-th token in the sentence. Formally, given $\mathbf{h}_i$, the final feature representation $\mathbf{h}'_i$ for each word is obtained through

$$\mathbf{h}'_i = (1 - \mathbf{c}_i) \odot \mathbf{h}'_{i-1} + \mathbf{c}_i \odot \mathbf{z}_i \tag{14}$$

where

$$\begin{aligned} \mathbf{c}_i &= \sigma(\mathbf{W}_c \mathbf{h}'_{i-1} + \mathbf{U}_c \mathbf{h}_i) \\ \mathbf{z}_i &= \tanh(\mathbf{W}_z(\mathbf{g}_i \odot \mathbf{h}'_{i-1}) + \mathbf{U}_z \mathbf{h}_i) \\ \mathbf{g}_i &= \sigma(\mathbf{W}_g \mathbf{h}'_{i-1} + \mathbf{U}_g \mathbf{h}_i) \end{aligned}$$

Here, $\mathbf{c}_i$ and $\mathbf{g}_i$ are control gates to decide the information flow. The final token-level prediction is made through

$$\hat{\mathbf{y}}_i = \text{softmax}\,(\mathbf{W}_l \cdot \mathbf{h}'_i + \mathbf{b}_l) \tag{15}$$

where $\mathbf{W}_l \in \mathbb{R}^{5 \times d'}$ transforms a d'-dimensional feature vector to class probabilities (note that we have five different classes as defined in Section 3.1). For unsupervised domain adaptation, we use shared parameters for both TRNN and GRU network in each domain, because the characteristic correlations between syntatic-related words or adjacent words are invariant across different domains.

For supervised single-domain setting, the joint model is trained on labeled training data in a specific domain, which is then evaluated on the test set in the same domain.

We use the cross-entropy loss as the training objective:

$$\ell_{\text{single}} = \sum_{i=1}^{n} \ell_y(\mathbf{y}_i, \hat{\mathbf{y}}_i) \tag{16}$$

where n is the total number of labeled training instances. For the cross-domain setting, to train the joint model with only labeled data $\mathcal{D}_S = \{(\mathbf{x}_i^S, \mathbf{y}_i^S)\}_{i=1}^{n_S}$ in the source domain, we generate another two sets of training data, including $\mathcal{D}_R = \{(\mathbf{r}_j, \mathbf{y}_j^R)\}_{j=1}^{n_S+n_T}$ for the auxiliary relation prediction task as well as $\mathcal{D}_d = \{(\mathbf{x}_m, \mathbf{r}_m, \mathbf{y}_m^d)\}_{m=1}^{n_S+n_T}$ for cDAN. Both $\mathcal{D}_R$ and $\mathcal{D}_d$ contain the combination of source and target training instances. The total loss is the combination of token-prediction loss ℓ_y, relation-prediction loss ℓ_R, and domain loss ℓ_d:

$$\ell_{\text{cross}} = \sum_{i=1}^{n_S} \ell_y(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \lambda \sum_{j=1}^{n_S+n_T} \ell_R(\mathbf{y}_j^R, \hat{\mathbf{y}}_j^R) - \gamma \sum_{m=1}^{n_S+n_T} \ell_d(\mathbf{y}_m^d, \hat{\mathbf{y}}_m^d) \tag{17}$$

where $\hat{\mathbf{y}}_i$ is the predicted extraction label in Equation (15), and ℓ_R is defined in Equation (10) for TRNN with auto-encoders or Equation (4) without auto-encoders. With cDAN, we adopt the Gradient Reversal Layer proposed by Ganin et al. (2016) to update the parameters. Specifically, we denote θ_f, θ_R, θ_d, and θ_y as parameters for feature learning, auxiliary relation prediction task, domain discriminator, and final prediction task, respectively. The Gradient Reversal Layer process will update the parameters as the following:

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial \ell_y}{\partial \theta_f} + \lambda \frac{\partial \ell_R}{\partial \theta_f} - \gamma \frac{\partial \ell_d}{\partial \theta_f} \right) \tag{18}$$

$$\theta_y \leftarrow \theta_y - \mu \left(\frac{\partial \ell_y}{\partial \theta_y} \right) \tag{19}$$

$$\theta_R \leftarrow \theta_R - \mu \left(\lambda \frac{\partial \ell_R}{\partial \theta_R} \right) \tag{20}$$

$$\theta_d \leftarrow \theta_d - \mu \left(\gamma \frac{\partial \ell_d}{\partial \theta_d} \right) \tag{21}$$

where μ is the learning rate, and λ and γ are trade-off parameters to control the impact of auxiliary loss and domain adversarial loss. The parameters for token-level predictions and relation-level predictions are updated jointly such that the information from the auxiliary task could be propagated to the target task to obtain better performance. This idea is in accordance with structural learning proposed by Ando and Zhang (2005), which shows that multiple related tasks are useful for finding the optimal hypothesis space. In our case, the set of multiple tasks includes the target terms extraction task and the auxiliary relation prediction task, which are closely related. The parameters are all shared across domains.

Table 1
Data statistics with number of sentences for each domain. w/l = with ground truth labels; w/o l = without labels.

Data set	Description	# Sentences	Training	Testing	Source	Target
R	Restaurant	5,841	4,381	1,460	4,381 (w/ l)	4,381 (w/o l)
L	Laptop	3,845	2,884	961	2,884 (w/ l)	2,884 (w/o l)
D	Device	3,836	2,877	959	2,877 (w/ l)	2,877 (w/o l)

6. Experiments

6.1 Data and Experimental Set-up

For experiments, we use benchmark customer reviews from three different domains, namely, restaurant, laptop, and digital devices. The data from the restaurant domain contains a combination of restaurant reviews from SemEval 2014 task 4 subtask 1 (Pontiki et al. 2014) and SemEval 2015 task 12 subtask 1 (Pontiki et al. 2015). For the laptop the domain, we extract laptop reviews from the laptop domain in SemEval 2014 task 4 subtask 1. The domain of digital device consists of customer reviews from Hu and Liu (2004), which include sentences from five digital devices. The statistics for each domain are shown in Table 1 with the number of sentences. To make robust comparisons, we conduct each experiment for three times and take the average performance as the final result. Specifically, we make three random splits to partition the data in each domain into a training set and a testing set with the proportion being 3:1. For each split, the training data are used to train the model, which is then evaluated on the test set. The numbers of sentences for both training and testing after each split are also shown in Table 1. For the single-domain problem, we use the labeled training data from the specified domain to train our model. For unsupervised domain adaptation, the training corpus consists of both source and target training data, but we only use the ground-truth labels from the source domain and ignore all the labels from the target domain in each transfer experiment. As shown in Table 1, when selected as the source domain, the training set with ground-truth labels (shown as "w/ l") is used, and the training set without labels (shown as "w/o l") is used for the target domain. For a complete evaluation, in each cross-domain experiment, we conduct both inductive and transductive testings. The inductive results are obtained using the test data from the target domain, and the transductive testing evaluates the model on the (unlabeled) training data from the target domain. We use F1 score for evaluation. Following the setting from existing work, only exact match could be counted as correct.

To conduct the experiments, we use the Stanford Dependency Parser (Klein and Manning 2003) to generate dependency tree for each sentence. The input for the whole network is the pretrained word embeddings obtained using word2vec (Mikolov et al. 2013), which is trained on 3M reviews from the Yelp data set[1] and electronics data set in Amazon reviews[2] (McAuley et al. 2015). We set the dimension of word embeddings as 100. The dimension of final features after GRU network is 50, and the context window

1 http://www.yelp.com/dataset_challenge.
2 http://jmcauley.ucsd.edu/data/amazon/links.html.

size is 3 for input feature vectors of GRU. Because of the relatively small size of the training data compared with the number of parameters, we first pre-train SRNN/TRNN without GRU on top for five epochs. The best pre trained model is selected to further train the joint model. For single-domain experiments, we use mini-batch with batch size 25 to train our model. Adaptive learning rate is adopted that is initialized at 0.02. For cross-domain experiments, mini-batch size is set as 30 and the rmsprop training strategy is used with the learning rate initialized at 0.01 for pre-training, and 0.001 for joint training. The whole corpus contains 43 different dependency relations, which is relatively large for the auxiliary task. As mentioned in Section 5.2.3, we cluster the dependency relations into a small number of relation groups in an unsupervised manner. The number of groups is set to be 20. The trade-off parameters for the auxiliary loss α and β are set as 1 and 0.001, respectively. The trade-off parameters for the total loss λ and γ are both set as 0.1. All the hyper-parameters for cross-domain experiments are selected based on three-round random-split validation on the extraction task in the source domain and relation prediction task over both source and target domain. Specifically, for source-domain validation data, we evaluate the model's performance on both terms, extraction task and relation prediction task. For target-domain validation data, we evaluate the performance on relation prediction task.

6.2 Results for Single-Domain Experiments

For meaningful performance comparisons, some typical baselines are selected:

- **CRF-1**: A traditional linear-chain CRF consisting of standard lexical features, including word string (within window-size 5), stylistics, POS tag (within window-size 5), and so forth.
- **CRF-2**: An extension of CRF-1 to include syntactic features, including dependency relations, head word, child words, and so on.
- **LSTM**: A recurrent neural network proposed by Liu, Joty, and Meng (2015). We use the same pretrained word embeddings as ours for fair comparison.
- **CRF-embedding**: Apply the standard linear-chain CRF with pretrained word embeddings as the input features. We use the same word embeddings as our proposed models.
- **SRNN**: The single-domain recursive neural networks without any sequential model on top. We apply softmax prediction directly on the output vectors from RNN.
- **RNCRF**: The proposed model in our preliminary work (Wang et al. 2016) without any mannually designed features. The joint model consists of an RNN and a CRF on top of the RNN.
- **SRNN-GRU**: The proposed joint model for single-domain aspect and opinion terms extraction. Different from RNCRF, SRNN-GRU replaces CRF at the topmost layer with GRU.

The comparison results in terms of average F1 scores are shown in Table 2. The first three rows are representative baseline models for sequence tagging problems. Both CRF-1 and CRF-2 take discrete human-engineered features to train the classifier.

Table 2
Comparisons with different baselines for single-domain aspect/opinion terms extraction.

Models	R		L		D	
	AS	OP	AS	OP	AS	OP
CRF-1	67.50	73.49	67.38	72.07	36.72	55.44
CRF-2	70.10	73.76	68.30	71.36	39.18	57.67
LSTM	74.48	77.05	72.48	73.05	**50.52**	63.14
CRF-embedding	72.10	75.94	70.22	73.56	46.78	53.44
SRNN	68.15	68.08	64.10	62.37	38.53	50.83
RNCRF	**76.78**	79.20	72.28	**73.74**	49.40	63.77
SRNN-GRU	75.70	**79.34**	**73.62**	73.31	50.28	**64.15**

Compared with deep models, the CRF models show inferior performance, demonstrating the advantage of continuous high-level features learned from deep neural networks. The linear combination of discrete features in CRF fails to capture high-level interactions inherent in the feature space. The dependency information enhances the prediction results when feeding into the CRF model. This demonstrates the effect of syntactic relations for the extraction task. LSTM greatly outperforms the CRF models, although it still falls behind our proposed model. The last four rows in Table 2 are variations of the proposed model. Specifically, CRF-embedding removes RNN for syntactic modeling, whereas SRNN removes the sequential layer on top. Without integrating both syntactic and sequential modules, the performance of CRF-embedding and SRNN are much worse than the joint model, which demonstrates the importance of both components for aspect and opinion terms extraction. The RNCRF model performs comparably with SRNN-GRU. This indicates that both CRF and GRU are effective for modeling sequential interactions.

6.3 Results for Cross-Domain Experiments

We compared the proposed model TRNN-GRU with several baselines for cross-domain aspect and opinion terms extraction. The baseline models are listed in the following:

- **RNCRF**: A joint model of a recursive neural network and CRF proposed by Wang et al. (2016) for single-domain aspect and opinion terms extraction. We make all the parameters shared across domains for target prediction.
- **RNGRU**: A joint model of RNN and GRU. The hidden layer of RNN is taken as input for GRU. This is similar to RNCRF by replacing CRF with GRU. The parameters are all shared across domains.
- **CrossCRF**: A linear-chain CRF with hand-engineered non-lexical features that are useful for cross-domain settings (Jakob and Gurevych 2010), for example, POS tags and dependency relations.
- **RAP**: The Relational Adaptive bootstraPping method proposed by Li et al. (2012) that uses TrAdaBoost to expand lexicons through common opinion terms and dependency relations.

- **Hier-Joint**: A cross-domain recurrent neural network proposed by Ding, Yu, and Jiang (2017) for aspect terms extraction across domains. The model integrates auxiliary tasks that are composed from manually defined rules.
- **ARNN-GRU**: The proposed joint model without cDAN, which was proposed in our previous work (Wang and Pan 2018), that combines a dependency-tree-based recursive neural network with GRU. An auto-encoder is incorporated in the auxiliary task to reduce label noise.
- **TRNN-GRU**: Based on ARNN-GRU, we integrate a conditional domain adversarial network that takes both word features and parent relation group as input to transfer knowledge across domains.

Note that we do not implement other recent deep adaptation models for comparison (Chen et al. 2012; Yongxin and Timothy M. 2015), because Hier-Joint (Ding, Yu, and Jiang 2017) has already demonstrated better performance than these models. The overall comparison results with the baseline models are shown in Table 3 with average F1 scores and standard deviations shown in brackets over three random splits. The results shown are based on the inductive setting, which evaluates on the test data in the target domain. Clearly, the results for aspect terms (AS) transfer are much lower than opinion terms (OP) transfer, which indicates that the aspect terms are usually disjoint across domains and are difficult to be adapted, whereas the opinion terms are easier to be aligned across domains. Hence the ability to transfer aspect knowledge from the source domain to the target domain becomes more crucial. For more challenging knowledge transfer in aspect terms, the proposed transferable recursive neural network shows substantial advantage over other baselines. We can see that both ARNN-GRU and TRNN-GRU achieve large performance gains for aspect extraction (AS), for example, 6.77%, 6.65%, and 11.07% improvement over the best-performing baselines for aspect extraction in R→L, L→D, and D→L, respectively. Without any domain adaptation strategies, RNCRF and RNGRU demonstrate inferior results by simply adopting the model trained on the source domain. This indicates the effectiveness of the structural correspondences built

Table 3
Comparisons with different baselines.

Models	R→L		R→D		L→R		L→D		D→R		D→L	
	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP
CrossCRF	19.72 (1.82)	59.20 (1.34)	21.07 (0.44)	52.05 (1.67)	28.19 (0.58)	65.52 (0.89)	29.96 (1.69)	56.17 (1.49)	6.59 (0.49)	39.38 (3.06)	24.22 (2.54)	46.67 (2.43)
RAP	25.92 (2.75)	62.72 (0.49)	22.63 (0.52)	54.44 (2.20)	46.90 (1.64)	67.98 (1.05)	34.54 (0.64)	54.25 (1.65)	45.44 (1.61)	60.67 (2.15)	28.22 (2.42)	59.79 (4.18)
Hier-Joint	33.66 (1.47)	- -	33.20 (0.52)	- -	48.10 (1.45)	- -	31.25 (0.49)	- -	47.97 (0.46)	- -	34.74 (2.27)	-
RNCRF	24.26 (3.97)	60.86 (3.35)	24.31 (2.57)	51.28 (1.78)	40.88 (2.09)	66.50 (1.48)	31.52 (1.40)	55.85 (1.09)	34.59 (1.34)	63.89 (1.59)	40.59 (0.80)	60.17 (1.20)
RNGRU	24.23 (2.41)	60.65 (1.04)	20.49 (2.68)	52.28 (2.69)	39.78 (0.61)	62.99 (0.95)	32.51 (1.12)	52.24 (2.37)	38.15 (2.82)	64.21 (1.11)	39.44 (2.79)	60.85 (1.25)
ARNN-GRU	**40.43** (0.96)	**65.85** (1.50)	35.10 (0.62)	60.17 (0.75)	52.91 (1.82)	72.51 (1.03)	40.42 (0.70)	**61.15** (0.60)	48.36 (1.14)	73.75 (1.76)	51.14 (1.68)	**71.18** (1.58)
TRNN-GRU	40.15 (0.77)	65.63 (1.01)	**37.33** (0.90)	**60.32** (0.66)	**53.78** (0.91)	**73.40** (0.45)	**41.19** (1.06)	60.20 (1.56)	**51.17** (0.99)	**74.37** (1.03)	**51.66** (1.27)	68.79 (1.63)

using an automatically generated auxiliary task as well as a domain discriminator to learn domain-invariant features.

Besides the pairwise transfer setting, we also conduct multisource adaptation experiments. Specifically, we select two different domains as source domains with labeled training data and the third domain as the target domain. Under this setting, two transfer strategies could be applied: The first strategy mixes the two source domains as a single domain, which reduces to the pairwise transfer problem. The second strategy is similar to ensemble learning where each of the two source domains is paired with the target domain to train a separate model. The final result is produced as a weighted average of the predictions made by those separate models. The F1 scores for multisource adaptation are the following (the best result among 2 strategies): 51.35/73.24 for aspect/opinion terms extraction when transferring from the laptop and the device domains to the restaurant domain, 46.77/68.96 for aspect/opinion terms extraction when transferring from the restaurant and the device domains to the laptop domain, and 41.34/62.10 for aspect/opinion terms extraction when transferring from the restaurant and the laptop domains to the device domain. Most of the results are not improved for aspect/opinion terms extraction compared with the single-source transfer setting. We conjecture that the reason behind this is the incompatibility among different source domains that leads to negative transfer effect. More empirical or theoretical studies on the multisource transfer setting will be conducted in our future work.

6.4 Transfer Analysis

6.4.1 Component Analysis. TRNN-GRU consists of several components that are crucial for achieving promising results. To investigate the effect of each component, we break down the joint model to generate a few variants of the proposed model as follows:

- **TRNN*-GRU**: Remove relation group clustering for cDAN in the original proposed model. In this case, the input for cDAN becomes the concatenation of hidden representation for each word and a one-hot feature vector indicating the index of the exact dependency relation.
- **TRNN^{-c}-GRU**: Remove conditional relation features for cDAN in TRNN-GRU. The input for the domain discriminator becomes the hidden vector for each word.
- **DRNN-GRU**: Remove the auxiliary task for relation predictions. The model can be reduced to RNGRU, similar to Wang et al. (2016), with a cDAN on top of the hidden representation of each word.
- **ARNN-GRU**: Remove the cDAN model, but only keep the auxiliary task with the auto-encoder. The resultant model is the same as the one we proposed in Wang and Pan (2018).
- **ARNN*-GRU**: Based on ARNN-GRU, remove the auto-encoder for clustering the dependency relations. The auxiliary task becomes a 43-class classification problem to predict each exact dependency relation.
- **TRNN**: Remove the sequence labeling model on the top, but use the hidden representation generated from the recursive neural network for the final prediction.

Table 4
Comparisons on each component of TRNN-GRU.

		R→L		R→D		L→R		L→D		D→R		D→L	
		AS	OP	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP
OUT	**TRNN-GRU**	40.15	65.63	**37.33**	**60.32**	**53.78**	73.40	**41.19**	60.20	**51.17**	**74.37**	**51.66**	68.79
	TRNN*-GRU	38.56	63.70	36.09	60.03	50.61	**73.68**	41.06	58.24	49.38	74.06	50.25	67.20
	TRNN^{-c}-GRU	**41.06**	**66.46**	34.71	58.77	53.45	72.89	40.07	59.20	49.03	72.78	51.06	69.82
	DRNN-GRU	40.68	64.88	35.45	60.30	48.58	72.38	40.23	59.31	50.30	73.41	46.57	67.45
	ARNN-GRU	40.43	65.85	35.10	60.17	52.91	72.51	40.42	**61.15**	48.36	73.75	51.14	**71.18**
	ARNN*-GRU	37.77	62.35	33.02	57.54	53.18	71.44	35.65	60.02	49.62	69.42	45.92	63.85
	TRNN	33.13	65.54	24.19	59.87	28.59	73.11	35.23	57.73	36.38	74.27	45.16	66.29
IN	**TRNN-GRU**	37.73	65.42	**35.42**	**60.55**	**52.79**	**73.42**	**39.85**	58.33	**50.13**	**73.72**	49.11	70.00
	TRNN*-GRU	37.35	65.02	34.64	59.76	49.57	72.40	39.24	57.80	48.72	74.41	48.74	68.92
	TRNN^{-c}-GRU	40.39	65.25	33.02	59.30	52.20	72.97	39.66	58.82	49.09	73.32	49.54	69.82
	DRNN-GRU	40.69	64.87	33.59	60.38	46.93	72.62	39.09	57.87	49.64	73.59	44.23	66.89
	ARNN-GRU	**41.27**	**65.44**	33.58	60.28	52.48	72.10	39.73	**60.18**	47.10	72.19	**50.23**	**70.21**
	ARNN*-GRU	39.07	62.80	31.86	57.81	52.51	71.67	35.74	59.72	49.48	69.36	45.69	64.82
	TRNN	31.16	65.22	27.53	59.88	27.68	72.95	36.18	58.25	36.99	72.90	47.69	65.95

For a more complete comparison, we conduct the experiments with both transductive setting, denoted as IN, and inductive setting, denoted as OUT in Table 4. The results are the average F1 scores among three splits. By observing similar performance for inductive and transductive experiments, the robustness of the model can be proved, that is, the proposed model indeed transfers knowledge from the source domain to the target domain even when test data in the target domain are not presented during training. From Table 4, we see that the joint model TRNN-GRU achieves the best performance most of the time. TRNN*-GRU shows inferior results for all except one experiment compared with TRNN-GRU. Similary, ARNN*-GRU deviates from ARNN-GRU with a large gap. These two comparisons indicate the importance of the auto-encoder for converting explicit dependency relations to their inherent relation groups in order to reduce the negative effect brought by inaccurate parsers. We also show that cDAN is advantageous by conditioning the domain discriminator on syntactic relations because TRNN^{-c}-GRU produces inferior results when removing the relation label features. The effect of the auxiliary task can be proved by the performance gap between DRNN-GRU and the final model. By removing the auxiliary component, DRNN-GRU is slightly worse than TRNN-GRU for R→D, L→D, and D→R, but deteriorates greatly for aspect extraction on L→R and D→L. On the other hand, the cDAN is also crucial, by observing the relatively low performance of ARNN-GRU compared with TRNN-GRU on most experiments. The results also indicate that the auxiliary task is more effective for knowledge transfer, compared with cDAN. These two components compensate for each other in the proposed joint model to achieve the best performance. Finally, the results for TRNN by removing GRU are much lower than the joint model, which proves the importance of combining syntactic tree structure with sequential modeling.

Although Table 4 shows that the auto-encoder is advantageous for cross-domain extraction compared with the model without it, it is still unclear whether the auto-encoder indeed reduces label noise brought by inaccurate dependency parsers. To clarify that, we construct another data set to simulate noisy dependency relations that are common for informal texts. Specifically, in the source domain of each transfer experiment, for each relation that connects to any aspect or opinion word, it has 0.5 probability of being replaced by any other relation. As shown in Table 5, we denote the model trained on

Table 5
Comparisons with and without antoencoders on noisy dependency relations.

Models	R→L		R→D		L→R		L→D		D→R		D→L	
	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP	AS	OP
ARNN*-GRU	37.77	62.35	33.02	57.54	53.18	71.44	35.65	60.02	49.62	69.42	45.92	63.85
ARNN*-GRU (r)	−4.80	−12.17	−6.81	−3.96	−17.30	−5.71	−2.78	−2.45	−9.59	−2.08	−5.86	−4.67
ARNN-GRU	40.43	65.85	35.10	60.17	52.91	72.51	40.42	61.15	48.36	73.75	51.14	71.18
ARNN-GRU (r)	**−1.16**	**−6.44**	**−1.68**	**−2.93**	**−7.12**	**−2.55**	**−2.21**	**−2.03**	**−3.00**	**−0.91**	**−0.69**	**−3.13**
TRNN*-GRU	38.56	63.70	36.09	60.03	50.61	73.68	41.06	58.24	49.38	74.06	50.25	67.20
TRNN*-GRU (r)	−5.93	−8.78	−3.19	−2.84	−7.38	**−1.16**	−2.67	−0.66	−4.20	−1.29	−10.69	−5.42
TRNN-GRU	40.15	65.63	37.33	60.32	53.78	73.40	41.19	60.20	51.17	74.37	51.66	68.79
TRNN-GRU (r)	**−4.90**	**−5.95**	**−0.73**	**−2.20**	**−5.50**	− 1.56	**−1.57**	**−0.12**	**−3.31**	**−0.94**	**−4.95**	−4.93

the newly constructed data set with noisy relations as (r). The results for models on the original data set (without (r)) are shown in F1 scores, whereas the results for their variants are presented as the reduction in terms of F1 performance compared to the original data set. For example, for the case of ARNN*-GRU, Table 5 shows −4.80 in the aspect extraction for R→L. This indicates ARNN*-GRU produces 4.80% lower F1 scores when trained on noisy data set compared with the original data set. Clearly, ARNN-GRU gives much better results than ARNN*-GRU on noisy training data, with less than 3.50% drop for most of the experiments. This demonstrates the effectiveness of an auto-encoder on reducing the negative influences brought by noisy auxiliary labels. Similarly, the performance of TRNN-GRU without an auto-encoder produces larger deterioration when the conditioned relation features are very noisy (shown as TRNN*-GRU (r)). This proves that the auto-encoder makes the model more robust to label noise and helps to adapt the information more accurately to the target data. Note that a large drop for $L \rightarrow R$ in aspect extraction might be caused by a large portion of noisy replacements for this particular data, which makes it too hard to train a good classifier. This may not greatly influence opinion extraction, as shown, because the two domains usually share many common opinion terms. However, the significant difference in aspect terms makes the learning more dependent on common relations.

6.4.2 Adaptation Analysis. We also demonstrate how the model aligns the two different domains in the training process via Figure 7. We use a measure of domain distance to evaluate the effect of knowledge transfer. Specifically, the maximum mean discrepancy (MMD) (Gretton et al. 2012) is adopted to compute the distance between the source domain and the target domain. Because the model works on word-level predictions that treat each word as an instance, we take the average word features for each sentence as the input to compute the domain distance and use a mixed RBF kernel for the MMD mapping. Specifically, the mixed RBF kernel consists of three kernels with different kernel widths: 0.5, 0.025, and 0.0125. We evaluate two forms of features; one is the input word embedding and the other is the hidden representation for each word obtained from TRNN. We should expect that the MMD result for hidden representations across domains is relatively small compared with the word embeddings and is decreasing while training, which shows that TRNN indeed aligns two different domains. As shown in Figure 7, the triangle markers on the top trajectory represent MMD values for input word embeddings across different domains, and the circle markers in the bottom trajectory represent MMD values for the learned hidden features. Obviously, the distribution distance in terms of hidden features is much smaller compared with the one on input features for each transfer experiment. The MMD on hidden features

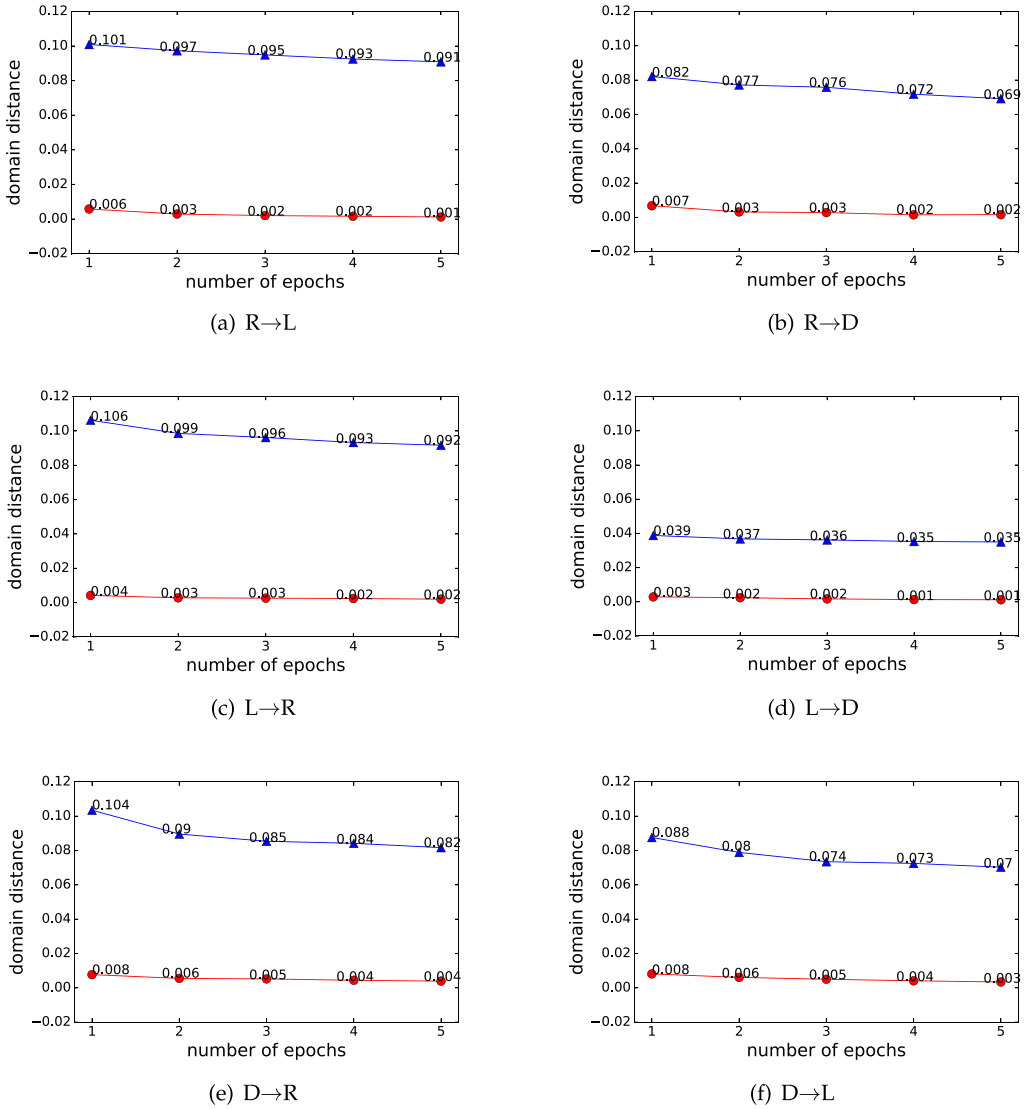


Figure 7
Analysis of domain distances for six different transfer experiments during training.

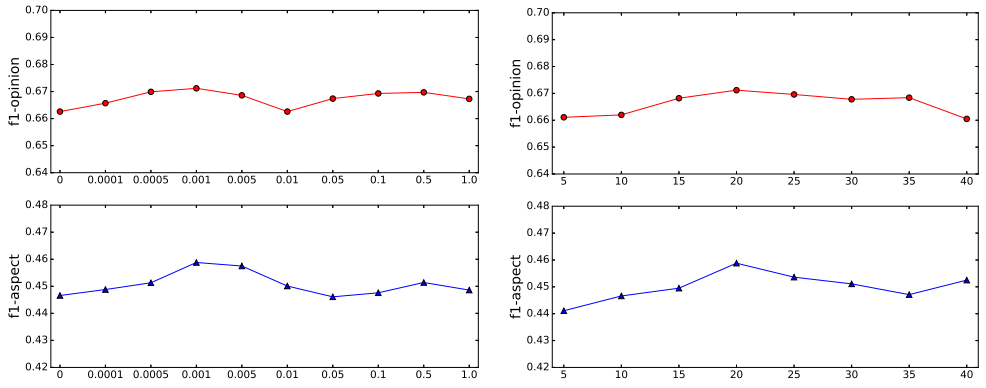
is also monotonically decreasing along the training process. This proves our claim that our model is able to transfer knowledge across different domains. Moreover, from the second and third rows of Figure 7, we notice that the MMD between the laptop and restaurant domains (or MMD between the device and restaurant domains) is much higher than the one between the laptop and device domains before applying TRNN. This follows the fact that the laptop domain and the device domain are much more similar because they both contain the reviews for the digital products. Despite the large discrepency on word features before training, our proposed model is able to map those input features that are far apart from two domains to closer proximity in the hidden space.

Table 6
Case studies on word and dependency relation clustering through auto-encoders.

G	Word	Dependency
1	this, the, their, my, here, it, I, our, not	det, poss, neg
2	quality, jukebox, maitre-d, sauces, portions, volume, friend, noodles, calamari	dobj, iobj, pobj
3	in, slightly, often, overall, regularly, since, back, much, ago	prep, mark, advmod
4	handy, tastier, white, salty, right, vibrant, first, ok	amod, advmod
5	get, went, impressed, had, try, said, recommended, call, love	ccomp, pcomp, xcomp
6	is, are, feels, believes, seems, like, will, would	cop, auxpass

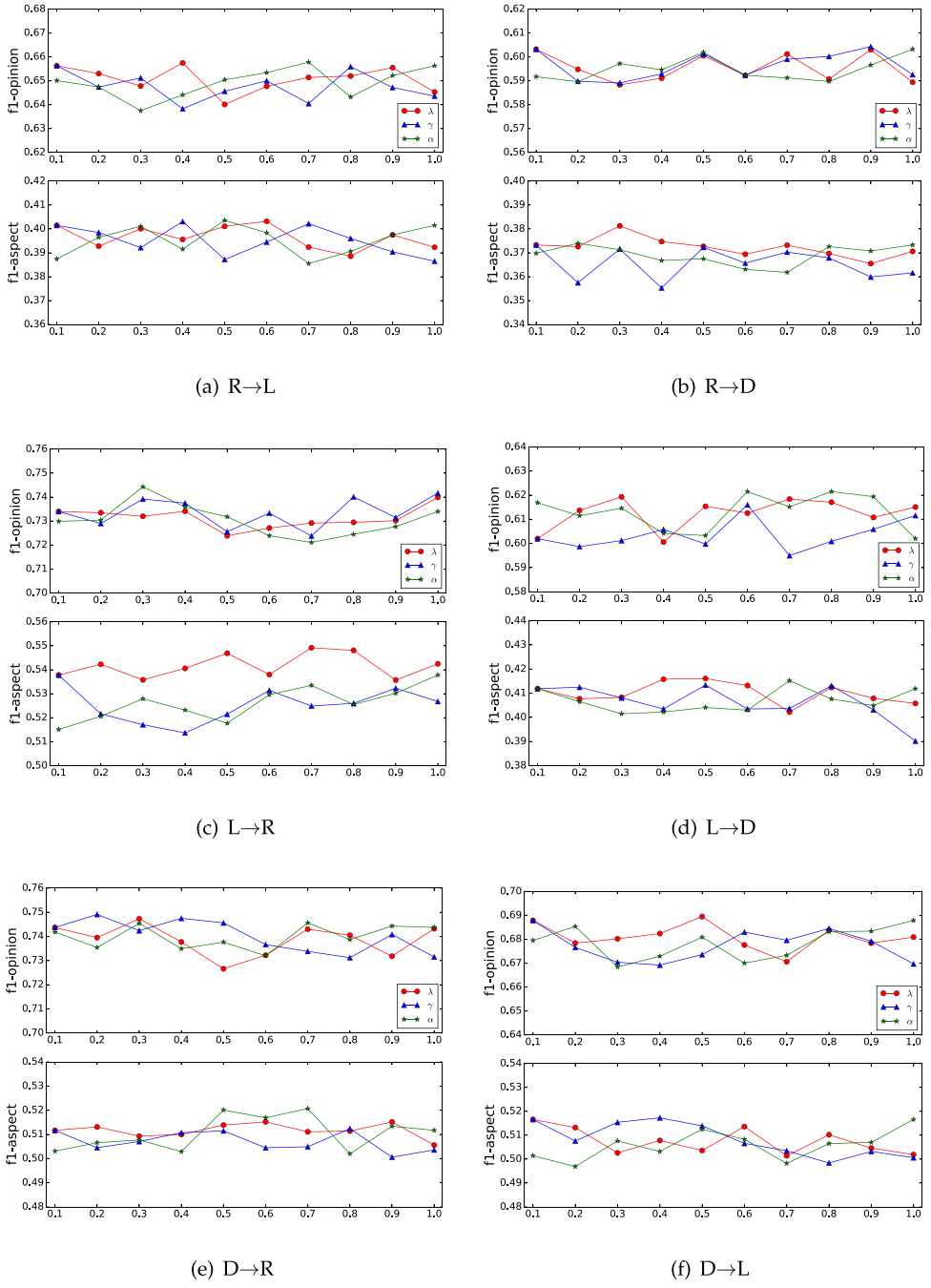
To qualitatively show the effect of the auto-encoders for clustering syntactically similar words across domains, we provide some case studies on the predicted groups of selected words in Table 6. Specifically, for each relation in the dependency tree, we use Equation (8) to obtain the most probable group to assign the word in the child node. This group is also used as the one-hot relation features for cDAN. The left column shows the predicted group index. The second column shows the corresponding words and the third column shows representative dependency relations within each specific group. Clearly, the words in the same group have similar syntactic functionalities, whereas the word types vary across groups. Dependency relations are also clustered according to their roles, for example, dobj and pobj both indicate that the child node is an object of the parent node. Moreover, the same dependency relation might be clustered into different groups according to different contexts—for example, advmod in some cases lies in the same group as amod when they both modify some target words, whereas advmod could also be clustered together with prep when describing some properties.

6.4.3 Robustness Analysis. We also demonstrate the robustness of our model via sensitivity tests. Figures 8 and 9 provide the performance variations by changing the hyperparameters. Specifically, Figure 8 provides the average performance over six source-target experiments for varying β and the number of clustered relation group $|G|$. Clearly, both experiments are relatively stable with less than 2% difference for aspect and opinion extraction. Moreover, Figure 8(a) also depicts the result when β is set to 0.



(a) β (b) $|G|$

Figure 8
Sensitivity studies for the average results on varying *beta* and the number of relation groups.



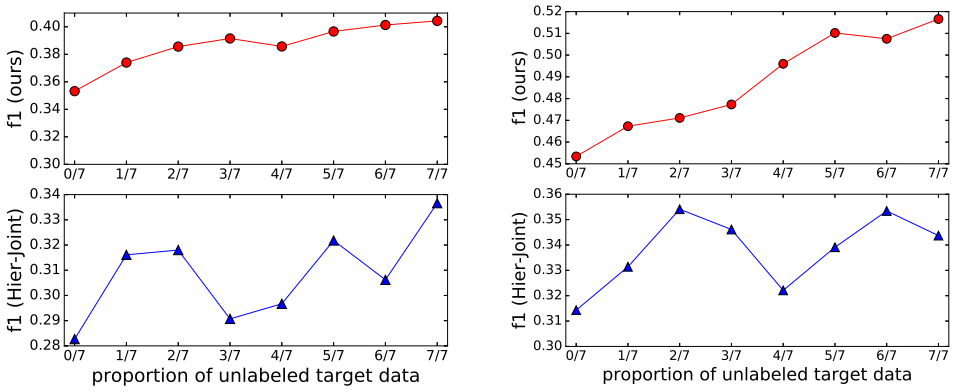
(a) R→L (b) R→D

(c) L→R (d) L→D

(e) D→R (f) D→L

Figure 9
Sensitivity studies for λ, γ, and α on six different transfer experiments.

As shown, when $\beta = 0$ (which indicates the removal of the orthogonality constraint for the auto-encoder loss), the performance drops slightly compared with $\beta \neq 0$. This demonstrates that the orthogonality constraint on relation clustering could



(a) F1-aspect on R→L (b) F1-aspect on D→L

Figure 10
F1 vs. proportion of unlabeled target data.

enhance the final prediction performance. Figure 9 demonstrates the sensitivity for λ, γ, and α. We can observe that when varying these parameters from 0.1 to 1.0, the performance for each source-target pair is stable with only small fluctuations. These two figures both demonstrate the robustness of our proposed model.

6.4.4 Learning Analysis. Furthermore, the ability of TRNN-GRU for knowledge transfer can be qualitatively shown in Figure 10. Specifically, we compare the results of TRNN-GRU with baseline model Hier-Joint on the performances with different proportions of unlabeled target training data from 0 to 1. Obviously, our model shows steady improvement with the increasing number of unlabeled target data for training. This pattern proves our model's capability of learning from target domain for adaptation.

7. Conclusion

In this article, we present the expressiveness and effectiveness of different forms of dependency-tree-based recursive neural networks for both single-domain and cross-domain settings. The deep recursive structure together with the dependency-tree information is able to associate automatic feature learning with syntactic structures, which have been proven to be crucial for both single-domain and cross-domain aspect/opinion terms extraction. The proposed models encode the syntactic interactions among the aspect and opinion words within each sentence for information propagation. To transfer knowledge across different domains, we integrate an auxiliary task on dependency relation prediction to build the structural correspondences for words between the source and target domains. At the same time, a conditional domain adversarial network is incorporated to learn domain-invariant word features based on their inherent syntactic structure. The proposed model can deal with noisy relation information effectively through an auto-encoder. Extensive experiments and analysis have been conducted to demonstrate the effect of each component of the proposed models quantitatively and qualitatively.

Acknowledgments

This work was supported by NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-2 grant MOE2016-T2-2-060, and a Singapore Lee Kuan Yew Postdoctoral Fellowship.

References

Ando, Rie Kubota and Tong Zhang. 2005. A framework for learning predictive struc tures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853.

Blitzer, John, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*, pages 187–205, Prague.

Bollegala, Danushka, Takanori Maehara, and Ken ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proceedings of ACL (1)*, pages 730–740, Beijing.

Chen, Minmin, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising auto-encoders for domain adaptation. In *Proceedings of ICML*, pages 1627–1634, Edinburgh.

Chen, Zhiyuan, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *Proceedings of ACL*, pages 347–358, Baltimore, MD.

Ding, Ying, Jianfei Yu, and Jing Jiang. 2017. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *Proceedings of AAAI*, pages 3436–3442, San Francisco, CA.

Dong, Li, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 49–54, Baltimore, MD.

Ganin, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of ICML*, pages 97–110, Bellevue, WA.

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773.

He, Ruidan, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of ACL*, pages 388–397, Vancouver.

Hu, Minqing and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*, pages 168–177, Seattle, WA.

Jakob, Niklas and Iryna Gurevych. 2010. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *Proceedings of EMNLP*, pages 1035–1045, Cambridge, MA.

Jiang, Long, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL*, HLT, pages 151–160, Portland, OR.

Jin, Wei and Hung Hay Ho. 2009. A novel lexicalized HMM-based learning framework for Web opinion mining. In *Proceedings of ICML*, pages 465–472, Montreal.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430, Sapporo.

Lakkaraju, Himabindu, Richard Socher, and Christopher D. Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*, pages 1–9.

Li, Fangtao, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of COLING*, pages 653–661, Beijing.

Li, Fangtao, Minlie Huang, and Xiaoyan Zhu. 2010. Sentiment analysis with global topics and local dependency. In *Proceedings of AAAI*, pages 1371–1376, Atlanta, GA.

Li, Fangtao, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. 2012. Cross-domain co-extraction of sentiment and topic lexicons. In *Proceedings of ACL*, pages 410–419, Jeju Island.

Li, Xin and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of EMNLP*, pages 2886–2892, Copenhagen.

Li, Zheng, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of IJCAI*, pages 2237–2243, Melbourne.

Liu, Bing. 2011. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, *Second Edition*. Data-Centric Systems and Applications. Springer.

Liu, Kang, Liheng Xu, Yang Liu, and Jun Zhao. 2013. Opinion target extraction using partially-supervised word alignment model. In *Proceedings of IJCAI*, pages 2134–2140, Beijing.

Liu, Kang, Liheng Xu, and Jun Zhao. 2012. Opinion target extraction using word-based translation model. In *Proceedings of EMNLP-CoNLL*, pages 1346–1356, Jeju Island.

Liu, Pengfei, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of EMNLP*, pages 1433–1443, Lisbon.

Long, Mingsheng, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. 2018. Conditional adversarial domain adaptation. In *NIPS*, pages 1647–1657, Montreal.

Lu, Yue, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of WWW*, pages 131–140, Madrid.

Ma, Dehong, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of IJCAI*, pages 4068–4074, Melbourne.

Ma, Tengfei and Xiaojun Wan. 2010. Opinion target extraction in Chinese news comments. In *Proceedings of COLING*, pages 782–790, Beijing.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester.

McAuley, Julian, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of SIGIR*, pages 43–52, Santiago.

Mei, Qiaozhu, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in Weblogs. In *Proceedings of WWW*, pages 171–180, Banff.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mirza, Mehdi and Simon Osindero. 2014. Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Pan, Sinno Jialin, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of WWW*, pages 751–760, Raleigh, NC.

Pang, Bo and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2), pages 1–135.

Pontiki, Maria, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of SemEval 2015*, pages 486–495, Denver, CO.

Pontiki, Maria, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of SemEval*, pages 27–35, Dublin.

Popescu, Ana Maria and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of EMNLP*, pages 339–346, Vancouver.

Qiu, Guang, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Reed, Scott E., Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training deep neural networks on noisy labels with bootstrapping. In *Workshop ICLR*, pages 1–11, San Diego, CA.

Titov, Ivan and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*, pages 111–120, Beijing.

Wang, Wenya and Sinno Jialin Pan. 2018. Recursive neural structural correspondence network for cross-domain aspect and opinion co-extraction. In *Proceedings of ACL*, pages 2171–2181, Melbourne.

Wang, Wenya, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 616–626, Austin, TX.

Wang, Wenya, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer tensor network for co-extraction of aspect and opinion terms. In *Proceedings of AAAI*, pages 3316–3322.

Wu, Yuanbin, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency

parsing for opinion mining. In *Proceedings of EMNLP*, pages 1533–1541, Singapore.

Xu, Hu, Bing Liu, Lei Shu, and Philip S. Yu. 2018. Double embeddings and CNN-based sequence labeling for aspect extraction. In *Proceedings of ACL*, pages 592–598, Melbourne.

Yin, Yichun, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of IJCAI*, pages 2979–2985, New York, NY.

Yongxin, Yang and Hospedales Timothy M. 2015. A unified perspective on multi-domain and multi-task learning. In *Proceedings of ICLR*, San Diego, CA.

Yu, Jianfei and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of EMNLP*, pages 236–246, Austin, TX.

Zhang, Lei, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. 2010. Extracting and ranking product features in opinion documents. In *Proceedings of COLING*, pages 1462–1470, Beijing.

Zhang, Meishan, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of EMNLP*, pages 612–621, Lisbon.

Zhao, Wayne Xin, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of EMNLP*, pages 56–65, Cambridge, MA.

Zhou, Guangyou, Zhao Zeng, Jimmy Xiangji Huang, and Tingting He. 2016. Transfer learning for cross-lingual sentiment classification with weakly shared deep neural networks. In *Proceedings of SIGIR*, pages 245–254, Pisa.