Deep interactive learning for fine-grained opinion mining : single-domain, cross-domain & cross-lingual

Wang, Wenya

2018

Wang, W. (2017). Deep interactive learning for fine-grained opinion mining : single-domain, cross-domain & cross-lingual. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/75872>

<https://doi.org/10.32657/10356/75872>

DEEP INTERACTIVE LEARNING FOR
FINE-GRAINED OPINION MINING:
Single-domain, Cross-domain & Cross-lingual

A dissertation submitted to
School of Computer Science and Engineering
Nanyang Technological University

by

Wenya Wang

in fulfilment of the requirements
for the degree of Doctor of Philosophy

April 2018

Abstract

Opinion mining or sentiment analysis provides a way to analyze the attitude or emotion of an opinion holder towards an entity or feature of any product. The emergence of opinion mining techniques comes along with the excessive volume of text data from social media made available to the public, such as product reviews, blog posts, forum discussions and Twitter website for social network.

Sentiment analysis has been actively studied among the Natural Language Processing (NLP) community since 2000. At this early stage, the main focus of sentiment analysis dealt with the overall sentiment polarity towards a given sentence or document, which we call coarse-grained sentiment analysis. For example, given a review sentence: “The sound is clean and clear”, the desired output should be **positive**, without knowing the exact object the reviewer is talking about. For this task, various machine learning classifiers and deep learning models were developed to learn opinion-bearing features for the whole sentence or document to make predictions. However, a mere sentiment score is far from enough to provide all the necessary information for decision making. That’s the reason why fine-grained opinion mining becomes more popular in a later stage.

Fine-grained opinion mining, or aspect-based sentiment analysis, mainly focuses on finding the opinion targets (aspects) from a given sentence and the emotions expressed towards them. This is more related to information extraction. Take the previous example review sentence, the model should be able to extract *sound* as the aspect and *clean and clear* as the opinion. The extracted information can be further used for aspect categorization, aspect sentiment classification and opinion summarization which provides a thorough analysis for the input. In this case, the task is much more difficult than coarse-grained sentiment classification because more fine-grained features are needed to make token-level predictions.

This thesis works on the task of aspect-based sentiment analysis to extract all the explicit aspects and opinions appeared in each review sentence. There has been active studies using machine learning models with manually-defined rules or extensive human-engineered features. However, this line of work required domain-specific knowledge and pipelined processes which is non-flexible for inconstant and informal texts. Another line of work applied deep learning to automatically learn the high-level representation for each word. However, they failed to model the important interactions among the words within a sentence. It has been shown that syntactic interactions and contextual relations among the words are especially crucial for information propagation. Bearing this in mind, this thesis designs some specific deep learning models that could automatically exploit the relationship among the input tokens in order to extract more accurate information. The models are flexible to learn the desired patterns without manual construction. Different from traditional deep learning models, the proposed approaches are able to capture fine-grained correlations among the input and could produce token-level representations inheriting the information from their correlated words.

Specifically, we begin with the task of supervised single-domain aspect and opinion terms extraction. The first model is a dependency-tree-based recursive neural network, which is able to exploit the interactions of syntactically related word pairs given the dependency tree of a sentence. This approach, however, still requires the dependency parser as a linguistic tool and is prone to the inevitable errors of the parser. This motivates the second method, which applies attention mechanism to model the interactions among the words automatically. We further extend from single-domain problem to cross-domain and cross-lingual settings, where labeled data only exists in the source domain (language), leading to an unsupervised adaptation problem. We explore deep models to learn shared spaces across different product domains within the same language as well as different languages. The shared spaces are built through common syntactic relations. We experiment all the above methods on benchmark datasets to prove the state-of-the-art performances of the proposed models.

Acknowledgments

My entire PhD studies and this dissertation would not have been possible without the generous support from the people around me. First of all and most importantly, I would like to express my sincere gratitude towards my supervisor Prof. Sinno Jialin Pan. Without his patient and insightful guidance and encouragement, I would not have been able to acquire the precious knowledge for deep learning and the NLP community. He guided me towards a scientific and critical way of thinking and doing research which is exceptionally beneficial for my PhD studies and will continue to influence my future researches. He gave me tireless one-on-one instructions on my research problems and at the same time provided me with enough freedom to learn what I am interested in. I am also very grateful to my co-supervisor Prof. Xiaokui Xiao for his kind advice and support.

For my four-year PhD life, I was fully sponsored by SAP Research & Innovation, Singapore, to which I would like to express my deep gratitude. This scholarship provided me valuable experience on how deep learning could be applied in the industries. I'm very grateful to Dr. Daniel Dahlmeier for being my mentor in SAP throughout my PhD career. His professional experience and advice helped me discover valuable research problems within deep learning and natural language processing. I enjoyed discussing with him on the up-to-date topics and methods in real-world applications. Thanks to my colleagues, Xin Zheng, Jin Wang, Ruidan He, Haodan Yang, Wei Cui, Qing Guo, Bingshui Da, Jianshu Li, Peng Ni, Kun Ouyang and Ray Han for all the support and generous sharing of the interesting topics.

I would like to thank to all my friends in Prof. Pan's group, Haiyan Yin, Hangwei Qian, Jianjun Zhao, Longkai Huang, Yu Chen, Shangyu Chen, Jianda Chen, Qiang Zhou and Jie Zhang for their constant companion and friendly support. I was so lucky to have

these many great people to work with. I am also truly thankful for the convenient space and excellent facilities provided by the Computational Intelligence Lab.

Of course, I would not be what I am now without the endless love from my family. They have provided me with all their support and encouragement throughout the time.

Contents

Abstract	i
Acknowledgements	iii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Background	1
1.2 Challenges	4
1.3 Approaches & Contributions	6
1.4 Dissertation Organization	9
2 Literature Review	11
2.1 Coarse-grained Sentiment Analysis	11
2.2 Fine-grained Opinion Mining	12
2.2.1 Aspect Terms and Opinion Terms Extraction	13
2.2.2 Aspect-level Categorization	14
2.2.3 Aspect-dependent Sentiment Classification	15
2.2.4 Deep Learning for Aspect-based Sentiment Analysis	16
2.3 Sentiment Analysis Across Domains & Languages	17
2.3.1 Adaptation in the Coarse-grained Setting	17
2.3.2 Adaptation in the Fine-grained Setting	18
2.4 Summary	19

3	Preliminary	21
3.1	Deep Learning Models for Input Interactions	21
3.1.1	Recurrent Neural Network	22
3.1.2	Recursive Neural Network	24
3.1.3	Attention-based Models	25
3.2	Problem Definition	27
4	Exploiting Token-level Interactions via Dependency Trees	29
4.1	Motivation	30
4.2	Introduction to Conditional Random Fields	31
4.3	Recursive Neural Conditional Random Fields	32
4.3.1	Dependency-tree Recursive Neural Network	32
4.3.2	Integration with CRFs	34
4.3.3	Joint Training for RNCRF	36
4.3.4	Advantage of RNCRF	37
4.4	Experiment	39
4.4.1	Dataset & Experimental Setup	39
4.4.2	Result & Analysis	41
4.5	Summary	45
5	Modeling Token-level Interactions via Attentions	47
5.1	Multi-task Learning	49
5.2	Problem Definition & Motivation	50
5.3	Memory Network with Coupled Attentions for Extraction	51
5.3.1	Attention with Tensor Operator	52
5.3.2	Coupled Attentions for Dual Propagation	56
5.3.3	Memory Network	56
5.4	Multi-task Memory Network for Extraction & Categorization	58
5.4.1	Category-specific MNCA	58
5.4.2	Shared Tensor Decomposition	59
5.4.3	Context-aware Multi-task Feature Learning	61
5.4.4	Auxiliary Task	62

5.5	Experiment	64
5.5.1	Dataset & Experimental Setup	64
5.5.2	Result & Analysis for Aspect and Opinion Terms Extraction . . .	65
5.5.3	Result & Analysis for Extraction and Categorization	69
5.6	Summary	74
6	Domain Adaptation via Syntactic Correspondences	75
6.1	Problem Definition & Motivation	77
6.2	Methodology	79
6.2.1	Recursive Neural Structural Correspondence Network	80
6.2.2	Reduce Label Noise with Auto-encoders	82
6.2.3	Joint Models for Sequence Labeling	84
6.2.4	Training	85
6.3	Experiments	86
6.3.1	Dataset & Experimental Setup	86
6.3.2	Result & Analysis	88
6.4	Summary	93
7	Language Adaptation via Transition-based Adversarial Network	95
7.1	Problem Definition & Motivation	97
7.2	Methodology	99
7.2.1	Transition-based Model	99
7.2.2	Transition-based Adversarial Network	101
7.2.3	Transition-based Adversarial Network with Syntactic Relations . .	103
7.3	Experiments	106
7.3.1	Dataset & Experimental Setup	106
7.3.2	Result & Analysis	107
7.4	Summary	113
8	Conclusions	115
	List of Publications	119

List of Submissions	121
References	123

List of Figures

3.1	The unrolled and rolled structure of RtNN.	22
3.2	An example of a constituency-tree-based RvNN.	24
3.3	Attention network for machine translation [3].	25
4.1	A linear-chain CRF.	31
4.2	Examples of a dependency tree and its corresponding DT-RvNN structure for a review sentence.	32
4.3	RNCRF structure for a review sentence.	34
4.4	An example for computing input-ouput potential for the second position <i>like</i>	35
4.5	Sensitivity studies on word embeddings.	44
5.1	A dependency example and illustration for the features of MNCA.	51
5.2	An example of second layer of MNCA.	53
5.3	Architecture of dual propagation memory networks for aspect & opinion terms extraction.	53
5.4	Independent attentions vs coupled attentions with tensor operator.	55
5.5	The architecture of each non-output layer used in MTMNs.	60
5.6	The architecture of the output layer used in MTMNs.	63
5.7	Visualization of attention weights for different tokens within a sequence.	69
5.8	Sensitivity studies for data S1 and S4.	69
6.1	An example of two reviews with similar syntactic patterns.	78
6.2	The architecture of RNSCN-GRU.	79
6.3	An autoencoder for relation grouping.	84

6.4	Sensitivity studies for $L \rightarrow D$	92
6.5	F1 vs proportion of unlabeled target data.	93
7.1	The overall architecture TAN.	99
7.2	An example of all the transition actions and states of each configuration of a sentence.	100
7.3	Configuration representation constructed from three recurrent neural net- works.	101
7.4	A configuration with proposed transition-based model.	103
7.5	F1 score vs percentage of unlabeled target training data.	111
7.6	Sensitivity tests for hyper-parameters.	112

List of Tables

1.1	Categorization of each proposed model.	8
4.1	SemEval Challenge 2014 task 4 dataset	39
4.2	Comparison results in terms of F1 scores.	42
4.3	Impact of different components.	43
5.1	Dataset description.	64
5.2	Aspect Categories for two domains.	65
5.3	Comparison results in terms of F_1 scores. AS (OS) refers to aspect (opinion) terms extraction.	66
5.4	Comparisons under varying layers and settings.	67
5.5	Prediction comparison between MNCA and RNCRF	69
5.6	Comparison results in terms of F_1 scores. ASC (OPC) refers to category-specific aspect (opinion) terms extraction. AS (OS) refers to aspect (opinion) terms extraction.	71
5.7	Comparison results with reductions.	72
5.8	Examples of attention scores for different categories.	73
6.1	Data statistics with number of sentences.	87
6.2	Comparisons with different baselines.	88
6.3	Comparisons with different variants of the proposed model.	90
6.4	Comparisons with different transfer setting.	90
6.5	Case studies on word clustering	92
7.1	Dataset statistics showing number of sentences	106
7.2	Comparisons with different baselines.	107

7.3	Comparisons with different variants of the proposed model.	110
7.4	Case studies on extracted aspect terms.	113

Chapter 1

Introduction

1.1 Background

Fine-grained Opinion Mining: In opinion mining, researches are mainly divided into two branches: coarse-grained sentiment analysis and fine-grained opinion mining. Coarse-grained sentiment analysis classifies an opinionated document or sentence as expressing an overall positive, neutral or negative opinion and assumes the given text containing only one opinionated target (entity). This assumption is usually not feasible for real-world applications, as people often comment on multiple aspects of a product. On the contrary, fine-grained opinion mining is able to provide a complete analysis on each appearing aspect in the opinionated text. Fine-grained opinion mining, also called aspect-based sentiment analysis, aims to extract the opinions expressed towards entities or features of entities, which we call aspects (or opinion targets). It consists of several subtasks such as aspect/opinion terms extraction, aspect categorization and aspect sentiment classification. When the given text contains multiple targets and opinions, e.g., “The dishes here taste fresh, but the service staff is unfriendly”, the task of extraction should identify “dishes” and “service staff” as aspect terms, then “fresh” and “unfriendly” as opinion terms. The task of categorization should classify “dishes” as category “FOOD” and “service staff” as category “SERVICE”. The task of sentiment classification aims to produce a positive score for “dishes” and a negative score for “service staff”.

Aspect-based sentiment analysis is much more challenging and complicated. Compared to coarse-grained sentiment analysis, where only sentence-level or document-level features are needed to predict the overall polarity, aspect-based tasks require specific modeling of fine-grained semantic and syntactic interactions within a input sentence. For example, the task of aspect and opinion terms extraction mainly depends on the ability to exploit the syntactic relatedness between the opinion target and the opinion expression. The surrounding context information around each word is also important to identify its possibility of being a target word. These two kinds of interactions could be illustrated using the previous example sentence. The first aspect term and opinion term correspond to “dishes” and “fresh”, respectively. The second aspect term and opinion term correspond to “service staff” and “unfriendly”, respectively. We should observe that the relationship between “dishes” and “fresh” should be similar or the same as the one between “staff” and “unfriendly” because both of the aspect words act as the subject of the opinion words. This explains the importance of syntactic relations for target terms extraction. To show the influence of the contextual interactions, we observe the second aspect term “service staff” which consists of two words. If “service” is deduced as an aspect word, it will indicate its subsequent word “staff” as an aspect word with high probability.

Deep Learning for NLP: Deep learning has become a popular and powerful machine learning technique that provides a way to learn multiple layers of abstractions automatically to represent the features of the input data and achieves state-of-the-art performances in various applications. In the NLP community, deep learning models are capable of extracting most useful and discriminative features that represent the semantic and syntactic information of the input texts. Compared to traditional machine learning with feature engineering, deep models save tremendous effort of designing and selecting language features which requires domain knowledge.

The development of deep learning for NLP begins with the distributed representations of words, which is also called word embeddings. Each word is associated with a continuous feature vector learned through a neural network [4, 87, 77]. There are two popular open-source tools for generating pre-trained word embeddings: **word2vec** [77] that is trained by predicting the target word given its contexts or vice versa, and **glove** [87] that is trained on a global word-word co-occurrence matrix. It has been shown that the learned word embeddings possess certain linguistic regularities and patterns [78] that are useful for various linguistic tasks. [20] proposed a general system to train word embeddings that are then fed into a neural network classifier for final predictions, e.g., POS tagging, chunking and semantic role labeling.

Based on the pre-trained word embeddings and linguistic structures, some deep learning models were developed to capture more semantically and syntactically related information. The first model is a recurrent neural network [29], which is specially designed for sequential inputs. The recurrent neural network makes the same process at each time stamp that takes the word of the current step and the hidden representation of the previous step as the input. This network is able to capture the information of the entire history when processing the current input and hence exploits context interactions. However, the formulation is prone to the problem of vanishing or exploding gradients. This motivates another formulation of recurrent neural network, which we call Long Short Term Memory (LSTM) [41]. LSTM deals with long-term dependencies through a memory cell that decides the amount of past information to memorize. Another variant of recurrent neural network is called Gated Recurrent Unit (GRU) [18]. Similar to LSTM, GRU can also learn long-term dependencies, and it has fewer parameters than LSTM which makes it easier to train. Recurrent neural networks have been widely applied and have shown promising results in sentiment analysis [107], machine translation [18], POS tagging [98] and Named Entity Recognition (NER) [58].

Attention mechanism was lately developed [3] that could be combined with recurrent neural networks to select relevant knowledge to be fed to the output. Different from traditional deep learning, attention mechanism aims to learn fine-grained focus on input texts in order to attend to most useful information. For example, [3] applied the attention model to selectively attend to the input tokens from the source language to produce the translation in another language. The model is able to focus on different part of the sentence during each step of the translation. Attention mechanism has also been used for summarization [97], syntactic parsing [115] and aspect-based sentiment classification [119].

Compared with recurrent neural network, recursive neural network is more general and aligned with the hierarchical structure of texts [101]. Indeed, a recursive neural network is a tree-structured network that combine words recursively to form phrases and sub-sentences. The network computes the representation for each node in a bottom-up manner that integrates all the information from its children. Recursive neural network has shown promising results in phrase representation and syntactic parsing [102], sentiment classification [104] and question answering [45].

1.2 Challenges

There has been growing interests in fine-grained opinion mining. Earlier works applied rule-based methods to extract aspect terms and opinion terms in an unsupervised manner. For example, [42] was the among the first to work on target terms extraction. They proposed to extract product aspects through association mining, and opinion terms by augmenting a seed opinion set using synonyms and antonyms in WordNet. In follow-up work, syntactic relations are further exploited for aspect/opinion extraction [91, 121, 94]. Among them, [94] used syntactic relations to double propagate and augment the sets

of aspects and opinions. With a set of seed collection of the target words, they iteratively expanded the set according to some pre-mined syntactic rules. Although the above models are unsupervised, they heavily depended on predefined rules and domain knowledge for extraction and are restricted to specific types of POS tags for product aspects and opinions, i.e., aspects are required to be nouns, and opinions are adjectives. The key limitations for their methods, however, are the lack of flexibility and error propagation because the pipelined procedure fails to send the information back to the previous steps. Another kind of unsupervised methods applied topic models for aspect extraction [17, 133]. This did not perform well for explicit extraction within each sentence, especially when an aspect contains multiple words.

In supervised settings, the task of target terms extraction has been treated as a sequence labeling problem, where each word is assigned a label. In this case, graphical models have been implemented to deal with structured output. [48, 61, 46, 71] proposed to use HMMs or CRFs to solve the problem. They treated the input sequence as a linear-chain structure and modeled the interactions among adjacent contexts for each word. These methods relied on rich hand-crafted features and did not consider interactions between aspect and opinion terms explicitly. Moreover, they took discrete features as the input that results in extremely high dimensions. The linear formulation of these graphical models are not able to exploit high-level interactions among the input. These limitations motivate the popularity of applying deep learning models for sequence tagging problems.

With deep learning, [66] applied various forms of recurrent neural networks on top of pre-trained word embeddings for aspect extraction. [125] proposed an unsupervised embedding method to encode dependency path into a recurrent neural network to learn high-level features for words, which are then taken as input features for CRFs for aspect extraction. Although these deep models could learn the high-level distributed representation for each word encoding the sequential context structure or the syntactic structure,

they failed to explicitly incorporate the linguistic structure within the deep model itself, so that the word-level label information could back-propagate to the syntactic structure learning process.

On the other hand, various deep learning models have been proposed for coarse-grained sentiment classification. These approaches focused on learning a sentence- or document-level representation that encodes the most relevant or obvious features within the input sequence, e.g., the feature of an opinion word. However, all the above methods are not directly applicable for the case of aspect/opinion extraction, because they are not able to exploit the fine-grained interactions among different units within the input sentence. The focus of our approaches is to learn the high-level representation of each word that encodes the useful information from other related words and their relationships for better information extraction.

1.3 Approaches & Contributions

To conquer all the limitations mentioned in Section 1.2, We propose a few deep models that are specially designed for fine-grained opinion mining. These models are end-to-end in terms of the training process, and do not require any pre-defined rules to guide the learning of the features. We modify popular deep models, e.g., recursive neural network and attention-based model to fit our fine-grained settings. Existing implementations of these models generate high-level representations in terms of increasing granularity. For example, a recursive neural network computes the features for higher-level nodes for a phrase by combining the features of its child words, and finally reach to the root node that represents the whole sentence by combining its sub-sentences. This is not helpful for word-level predictions, as the model treat each word as an independent unit. Instead of composing features with increasing granularity, We propose to generate representations in the same granularity (word level), but with increasing hierarchy of each word's

functionality. All the proposed models integrate the interactions and information propagations of aspect terms and opinion terms into the deep learning models naturally and intrinsically.

We observe that the development for fine-grained opinion mining has been slow and inattentive, compared to coarse-grained sentiment classification. There are two reasons that might cause the problem. The first reason is the difficulty in identifying the exact target words that may consists of a text span. It appears to be more challenging when it comes to informal text in social media, where grammatical rules are not strict. In this case, the existing methods depending on pre-defined rules are not flexible to extract the targets. This motivates our first two works by designing flexible and accurate deep learning models. The second reason is the limited resources that are available with fine-grained annotations, which are hard to obtain. To deal with this limitation, our next two works propose some unsupervised adaptation methods. Given labeled data in a resource-rich domain or language, the models are able to transfer knowledge to any target domain or language with no annotations in order to extract aspect/opinion terms.

To summarize, our proposed models achieve the state-of-the-art performances most of the times compared to baseline models on well-known benchmark datasets. The relationship among different proposed models is shown in Table 1.3. Specifically, Chapter 4 and Chapter 5 introduce our proposed models for supervised single-domain extraction and Chapter 6 and Chapter 7 discuss unsupervised adaptation methods across domains and languages, respectively. Regarding external resources, our first work requires a dependency parser to generate syntactic structures. Then the second model avoids this limitation through attention mechanism. However, for cross-domain and cross-lingual settings, the syntactic information is still required to build the correspondences across different feature spaces. Specific contributions of each work are illustrated in the following.

	single-domain	cross-domain	cross-lingual
with syntactic information	Chapter 4	Chapter 6	Chapter 7
without syntactic information	Chapter 5	future work	

Table 1.1: Categorization of each proposed model.

In the first work, we propose a novel joint model that integrates recursive neural networks and conditional random fields into a unified framework for explicit aspect and opinion terms co-extraction. The proposed model learns high-level discriminative features and double propagates information between aspect and opinion terms, simultaneously. It exploits both the syntactic relations among the words as well as the context information. Moreover, it is flexible to incorporate hand-crafted features into the proposed model to further boost its information extraction performance.

In the second work, we propose an end-to-end solution with attention mechanism that does not require any parsers or other linguistic resources for preprocessing. The model consists of a memory network that automatically learns the complicated interactions among aspect words and opinion words. Moreover, we extend the network with a multi-task perspective to solve a finer-grained problem, i.e., category-specific aspect and opinion terms extraction that involves both extraction and categorization of the extracted terms. To this end, we propose an end-to-end multi-task memory network, where aspect/opinion terms extraction for a specific category is considered as a task, and all the tasks are learned jointly by exploring commonalities and relationships among them.

In the third work, we extend the single-domain supervised information extraction problem to an unsupervised domain adaptation problem, where only unlabeled data is available in the target domain. We develop a novel recursive neural network that could reduce domain shift effectively in word level through syntactic relations. We treat these relations as invariant “pivot information” across domains to build structural correspondences and generate an auxiliary task to predict the relation between any two adjacent

words in the dependency tree. Moreover, to model the sequential context interaction, we develop a joint architecture that combines cross-domain recursive neural network with a sequential labeling model for cross-domain aspect and opinion terms extraction.

In the last work, a even more difficult task is being solved that requires transferring knowledge from a source language with labeled training data to a target language without any annotations. This involves adaptation among heterogeneous spaces which is more complicated than cross-domain information extraction. To achieve this task, traditional sequential labeling strategy is not applicable. Hence, we utilize an entirely different framework, a transition-based model that reads a word each time and forms a series of configurations that represent the status of the whole sentence. The model represents each configuration as a continuous feature vector and align these representations from different languages into a shared space through an adversarial network. In addition, syntactic structures are also integrated into the deep model to achieve more syntactically-sensitive adaptations.

Extensive experiments are conducted for each of the proposed model on several benchmark datasets. Most of the benchmark datasets come from SemEval challenges that provide complete fine-grained annotations for multiple domains and languages. The experimental results all show that these models achieve the state-of-the-art performances over existing baseline systems.

1.4 Dissertation Organization

This dissertation is carried out in the following organization. To begin with, Chapter 1 introduces the necessary background knowledge of the task at hand and the current models that are generally popular. It also states the limitations of the existing approaches, which motivate all the following work included in this thesis, with the general contributions discussed later on. The next chapter provides an overview of the existing approaches

proposed for fine-grained opinion mining, followed by Chapter 3 of preliminary that provides the description of three specific deep models that are being used as the base model in the following works. The subsequent chapters separately introduce each of the proposed model in more details with the model illustration as well as the experiments. In the end, Chapter 8 gives the final conclusion with several contributions of our works.

Chapter 2

Literature Review

This chapter provides an overview of the existing works for sentiment analysis, with extra focus on fine-grained opinion mining that are related to our study. In the first section, various related work about coarse-grained sentiment analysis are briefly introduced. This is then followed by an extensive description about different subtasks related to fine-grained opinion mining, including the task of aspect terms and opinion terms extraction, aspect-level categorization, aspect-dependent sentiment classification and a special attention on deep-learning-based fine-grained opinion mining. The next section further explores the methods beyond single-domain supervised learning strategies to unsupervised domain adaptation problems within sentiment analysis. A even more complicated problem involving transfer within heterogeneous spaces is surveyed that involves various methods for cross-lingual sentiment analysis.

2.1 Coarse-grained Sentiment Analysis

Coarse-grained sentiment analysis only deals with the task of assigning a polarity or sentiment score corresponding to the entire sentence or document. This task is analogous to the task of text classification, in which the feature representation of the entire input text is of great importance to convey the task-relevant information. Early works applied bag-of-words (BOW) strategies to construct input features, where each dimension

represents the frequencies or tf-idf of the corresponding word appeared in text. Then a classifier was applied on top, e.g., SVM or neural networks [81]. However, BOW representation is sparse and fails to consider ordering and semantic meanings. In this case, deep models, e.g., auto-encoders could be used to convert these BOW representations to high-level dense vectors in order to be fed into the final classifiers. [32, 129] applied auto-encoders to map from BOW vector to real-valued feature vector. [49] applied Convolutional neural network (CNN) as the mapping function. On the other hand, word embeddings could eliminate the need for BOW representations and directly generate a dense vector for each word that encodes semantic and syntactic characteristics. Several works were developed to take the word embeddings as the input and generate sentence- or document-level representations through CNN [108, 51], variants of recurrent neural networks [123], attentions and memory networks [27, 124].

Moreover, for sentence-level sentiment classification, additional linguistic information could be incorporated to learn a better discriminative model. Syntactic parse trees have been used to construct recursive neural networks that iteratively combine sub-parts of the sentence until reaching to the root node that represents the whole sentence [102, 103, 99, 104]. Dynamic CNN was proposed for learning the short and long-range relations [50]. Part-of-speech tags could also be encoded in the representation learning [43] to enhance the learned features. Moreover, sentiment lexicons could provide additional knowledge while learning [110, 93].

2.2 Fine-grained Opinion Mining

Coarse-grained sentiment analysis only provides an overall opinion score, ignoring the specific targets or aspects of the opinion. To consider more fine-grained information for structured sentiment analysis, many works have been carried to deal with aspect-based

sentiment tasks. This proves to be much more difficult than predicting a sentiment polarity, because a model is required to learn fine-grained interactions between the target term and its contexts. Existing works mainly covered three importance subtasks, namely aspect/opinion terms extraction, aspect-level categorization and aspect-dependent sentiment classification. Deep learning models have grown their interests in fine-grained opinion mining. The last part introduces existing deep models for this task.

2.2.1 Aspect Terms and Opinion Terms Extraction

There have been a number of works proposed for aspect/opinion terms extraction. Among the earliest works, Hu et al. [42] proposed to extract product aspects through association mining, and opinion terms by augmenting a seed opinion set using synonyms and antonyms in WordNet. In follow-up work, syntactic relations were further exploited for aspect/opinion extraction [91, 121, 94]. For example, Qiu et al. [94] used syntactic relations to double propagate and augment the sets of aspects and opinions. If an aspect word has been identified and it has a dependency relation with another word, where the relation and the word satisfy some pre-defined rules, this word could be extracted as an aspect/opinion word. All the above methods heavily depended on predefined rules for extraction and were restricted to specific types of POS tags for product aspects and opinions, e.g., aspects were restricted to be nouns and opinions were adjectives, making them inflexible. Topic models were also applied for aspect extraction [74, 17, 133, 111, 8, 62]. [74] used a mixture model based on both aspect and sentiment models. [111] proposed a multigrain topic model that could discover both global topics as well as local topics. [62] 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 not capable of learning multi-word expressions.

The above models were all learned in an unsupervised manner that achieved inferior results compared to supervised methods. For supervised learning, the task is treated as a sequence labeling problem. [48, 61, 46, 71, 19] proposed to use graphical models, such as HMMs or CRFs to learn the context interactions among the word sequence and predicted a label for each word. These methods relied on rich hand-crafted features and did not consider interactions between aspect and opinion terms explicitly. Another direction was to use word alignment model to capture opinion relations among a sentence [65, 64]. This method, however, required sufficient data for modeling desired relations. Recently, deep learning methods have been proposed for this task. [66] applied recurrent neural network on top of pre-trained word embeddings for aspect extraction. [125] proposed an unsupervised embedding method to encode dependency path into a recurrent neural network to learn high-level features for words, which were taken as input features for CRFs for aspect extraction.

2.2.2 Aspect-level Categorization

For aspect categorization, a set of categories is given in advance, e.g., environment could be an aspect category for restaurant reviews. The task has two variants: one interpretation is to classify each review to one of the pre-defined categories. Another interpretation is to group the pre-extracted aspect terms into different categories.

[37] applied multiple levels of LDA to group aspect words into more and more general concepts. Although topic models or clustering based approaches [112, 69, 133, 17, 105, 128, 16, 72] were able to group potential aspect terms into different clusters or topics, where topic modeling methods learn a distribution for each aspect category, however, they failed to align the semantic categories with their assigned groups. For grouping aspect terms, most existing methods assumed the aspect terms be extracted in advance, and aimed to predict their corresponding categories [10, 128, 130, 131]. [10, 128] applied

some similarity measures to group the aspects according to a taxonomy. To apply these methods, one needs to first identify aspect/opinion terms as a preprocessing step. For deep learning models, [38] proposed to use neural attentions to group reviews into different aspect categories, with the intuition similar to topic modeling. [57] used recursive neural networks to compute representations for phrases and sentences, which were then used as features for categorization.

2.2.3 Aspect-dependent Sentiment Classification

For aspect-dependent sentiment classification, an aspect is given and the task is to predict a sentiment polarity towards this aspect. A key component to learn the sentiment polarity of the aspect is the way to model its contextual influences and their interactions. Earlier works either applied supervised learning based on aspect-dependent features in the dependency trees [47], indicative words to segment a sentence [24], or unsupervised learning based on lexicon knowledge [42, 23]. [138] aggregated the opinions through pooling. [83] used pre-defined rules and domain knowledge to learn aspect-level sentiment polarities through constructed concept-level directed acyclic graph. [112, 118, 117] applied topic models to jointly learn aspect and rating distributions.

Deep models have also been actively applied for aspect-dependent sentiment analysis. [26] introduced Adaptive Recursive Neural Network (AdaRNN) for target-dependent twitter sentiment classification that learned to propagate the sentiment of the context words surrounding the target. [110] integrated rich features that were learned automatically from deep neural networks. [109] applied a memory network to selectively attend relevant context words given the target. [96] applied hierarchical LSTM network to model inter- and intra-sentence relations for target-dependent sentiment classification. [60] proposed to extract rationales for each target through auto-encoders. [70] proposed an Interactive Attention Network to learn attentions for both target and its contexts.

2.2.4 Deep Learning for Aspect-based Sentiment Analysis

Recent studies have shown that deep learning models can automatically learn the inherent semantic and syntactic information from data and thus achieve better performance for sentiment analysis [103, 99, 104, 32, 50, 51, 59]. These methods generally belonged to sentence-level or phrase/word-level sentiment polarity predictions. Regarding aspect-based sentiment analysis, most works were proposed for aspect-dependent sentiment classification, categorization or opinion expression extraction. As mentioned in Section 2.2.3, various deep models could learn the sentiment polarity of a given target aspect. They include recursive neural networks [26], recurrent neural networks [107, 96, 132], attentions [109, 70, 15] and auto-encoders [60]. For aspect categorization, attentions could also be used to learn aspect-relevant features in order to make predictions [38, 122]. Recursive neural network captured syntactic information for better categorization [57]. For opinion expression extraction, Irsoy et al. [44] applied deep bidirectional recurrent neural networks.

Nevertheless, there is little work in aspects and opinions co-extraction using deep learning models. To the best of our knowledge, the work of [66] and [125] were the most related to our subsequent works. [66] proposed to combine various forms of recurrent neural network and word embeddings to extract explicit aspect terms. However, the proposed model simply used recurrent neural network on top of word embeddings, and thus its performance heavily depended on the quality of word embeddings. In addition, it failed to explicitly model dependency relations or compositionality within certain syntactic structure in a sentence. [125] proposed an unsupervised learning method to improve word embeddings using dependency path embeddings. A CRF was then trained with the embeddings independently in the pipeline. However, this approach separated the feature learning process with the final prediction, which made it difficult to propagate the information within the network.

2.3 Sentiment Analysis Across Domains & Languages

Unsupervised sentiment analysis for a target domain or language with only labeled source domain or language data proves to be much more challenging than single-domain or single-language sentiment analysis. Because the model learned from the source data is not directly applicable to the target data due to considerable difference between the distributions of source domain (language) data and target domain (language) data, known as domain shift. The problem becomes exaggerated when it comes to cross-lingual sentiment analysis, because the source language and target language lie in heterogeneous spaces, making it hard to find alignments. Various methods have been proposed to address the adaptation problem in the case of coarse-grained sentiment classification. However, when it comes to fine-grained opinion mining, very few approaches have been proposed due to the difficulty in bridging the gap in a more fine-grained level, e.g., word distributions.

2.3.1 Adaptation in the Coarse-grained Setting

There have been a number of domain adaptation approaches proposed for coarse-grained sentiment classification. Among existing methods, one active line focused on projecting original feature spaces of two domains into the same low-dimensional space to reduce domain shift using pivot features as a bridge [5, 86, 7, 127]. Another line learned domain-invariant features via auto-encoders [32, 14, 135]. Our work is more related to the first line by utilizing pivot information to transfer knowledge across domains, but the proposed model integrates the idea into a unified deep architecture that can fully utilize syntactic structure for domain adaptation.

For cross-lingual sentiment classification, one approach is to use machine translation to translate into source or target language to build a classifier for target language [116, 85]. Another approach is to use parallel corpora or word pairs to learn shared features [92, 75,

134]. Bilingual representation learning has also been widely studied [53, 39, 13, 35, 137]. Moreover, [136] [136] used marginalized stack denoise auto-encoder to align document representations from heterogeneous spaces. [16] applied domain adversarial network [31] to train language-independent document representations.

2.3.2 Adaptation in the Fine-grained Setting

Different from domain adaptation for coarse-grained sentiment classification, the fine-grained problem appears to be much more challenging. Very few approaches exist for cross-domain fine-grained opinion analysis due to the difficulties in fine-grained adaptation. [63] proposed a bootstrap method based on the TrAdaBoost algorithm [21] to iteratively expand opinion and aspect lexicons in the target domain by exploiting source-domain labeled data and cross-domain common relations between aspect terms and opinion terms. However, their model required a seed opinion lexicon in the target domain and pre-mined syntactic patterns as a bridge. [25] proposed to use rules to generate auxiliary supervision on top of a recurrent neural network to learn domain-invariant hidden representation for each word. The performance highly depended on the quality of the manually defined rules and the prior knowledge of a sentiment lexicon. In addition, the recurrent structure failed to capture the syntactic interactions among words intrinsically for opinion extraction. The requirement for rules makes the above methods non-flexible.

However, fine-grained-level adaptation across languages has not been widely investigated yet. The reason might be the difficulty in knowledge transfer across heterogeneous input spaces for different languages, especially when the segmentations rules are different. In this thesis, we propose an effective deep model that can address all the above limitations for domain adaptation, as well as a new architecture designed for cross-lingual aspect terms extraction.

2.4 Summary

For single-domain sentiment analysis, a wide variety of models have been proposed for coarse-grained sentiment classification, which mainly focused on constructing overall sentence- or document-level representations, either through mapping from bag-of-word features or through integrating continuous word embeddings. Fine-grained opinion mining, however, appears to be much more challenging that requires specific modeling of the fine-grained interactions among each sentence. In this case, most existing methods depended on prior knowledge about the syntactic structure and a sentiment lexicon. Traditional machine learning techniques require extensive human-engineered features, which is inconvenient. To conquer this limitation, deep learning models were proposed to automatically learn the task-dependent high-level features. However, most of these models failed to exploit necessary correlations among the input tokens, which are crucial for final predictions. By extending single-domain to multi-domain settings, unsupervised domain adaptation techniques are desired to transfer knowledge from source domain to target domain. A key point is to build common knowledge between two domains to bridge the gap. This also applies to cross-lingual sentiment analysis, except that the cross-lingual problem requires adaptation across heterogeneous spaces. Existing models mostly deal with coarse-grained adaptation problem, but hardly examine fine-grained adaptation solutions, which motivates our current research.

Chapter 3

Preliminary

This chapter provides a detailed introduction to some deep learning models that are effective for various NLP tasks. Specifically, three models are presented, namely, recurrent neural network (RtNN), recursive neural network (RvNN) and attention mechanism. They are used as the base models for most of our following work, because specially designed constructions of these networks are beneficial to automatically exploit token-level interactions within each input sentence without additional resources and lexicons. After introducing the basic models, a formal definition of the problem we are trying to solve and its corresponding notations are given to facilitate the subsequent elaboration.

3.1 Deep Learning Models for Input Interactions

This section reviews three deep learning models that are popular for modeling text data. All of these networks are capable of modeling different kinds of compositions among the input tokens, i.e., sequential composition for RtNN, syntactic composition for RvNN and selective composition for attention models. The next sub-sections will separately elaborate each model in more details.

Figure 3.1: The unrolled and rolled structure of RtNN.

3.1.1 Recurrent Neural Network

Recurrent neural networks are introduced in 1990 by [29] to deal with sequential information. Different from traditional feed-forward neural networks, RtNN processes each input unit independently with the same procedure, where at each time stamp, the network takes the current input unit as well as the hidden state from the previous time stamp to produce the current hidden state. From this construction, the output at each time stamp is able to memorize all the history information beforehand.

The basic architecture of a RtNN is shown in Figure 3.1. The left part shows the unrolled unit of the recurrent mechanism and the right part rolls to the entire sequence. Formally speaking, given an input sequence $X = \{x_1, x_2, \dots, x_T\}$, a RtNN produces a hidden vector representation h_t for $x_t \in X$ through the following computation:

$$h_t = f(W_x x_t + W_h h_{t-1}), \quad (\text{Eq. 3.1})$$

where W_x and W_h are transformation matrices, f is a non-linear activation function. The output y_t for each time stamp can be obtained by

$$y_t = \text{softmax}(W_c h_t). \quad (\text{Eq. 3.2})$$

Here, y_t could be the POS tag for the corresponding word and W_c is a classification matrix. Obviously, this formulation makes RtNN suitable for modeling context interactions,

because the prediction at each position is affected by its previous contexts. The shared parameter at each time stamp indicates that RtNN performs the same task at each time.

The construction of the above network is prone to vanishing gradient or exploding gradient problem when the dependencies are too long. To overcome this problem, some variants of RtNN were proposed, e.g., Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). LSTM consists of a chain of repeating modules, where a module includes a memory cell and its gating variables to control the information flow [41]. Formally, the computation for LSTM replaces (Eq. 3.1) with the following procedures:

$$f_t = \sigma(U_f x_t + V_f h_{t-1}), \quad (\text{Eq. 3.3})$$

$$i_t = \sigma(U_i x_t + V_i h_{t-1}), \quad (\text{Eq. 3.4})$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(U_c x_t + V_c h_{t-1}), \quad (\text{Eq. 3.5})$$

$$o_t = \sigma(U_o x_t + V_o h_{t-1}), \quad (\text{Eq. 3.6})$$

$$h_t = o_t \cdot \tanh(c_t), \quad (\text{Eq. 3.7})$$

where f_t , i_t and o_t correspond to *forget gate*, *input gate* and *output gate*, respectively to control how much of the information from the history to be memorized. [18] proposed a simpler network called GRU that requires less parameter to be trained compared to LSTM. It combines *forget gate* and *input gate* into a single update gate, resulting in the following computation:

$$z_t = \sigma(U_z x_t + V_z h_{t-1}), \quad (\text{Eq. 3.8})$$

$$r_t = \sigma(U_r x_t + V_r h_{t-1}), \quad (\text{Eq. 3.9})$$

$$\tilde{h}_t = \tanh(U_h x_t + V_h (r_t \cdot h_{t-1})), \quad (\text{Eq. 3.10})$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t. \quad (\text{Eq. 3.11})$$

In the sequel, we will use f_{GRU} to represent the computations of (Eq. 3.8)-(Eq. 3.11).

Figure 3.2: An example of a constituency-tree-based RvNN.

3.1.2 Recursive Neural Network

Among deep learning methods, RvNN could be viewed as a generalization of RtNN that works on any directed acyclic graph structure, e.g., tree structures from the parsed sentences. RvNN has shown promising results on various NLP tasks, such as learning phrase representations [102], sentence-level sentiment analysis [104], language parsing [101], and question answering [45]. The structure proves to be more beneficial for NLP because of its ability to compose the representations of higher nodes from their lower constituents in the sentence hierarchy. The tree structures used for a RvNN have two formulations, namely constituency tree and dependency tree. In a constituency tree, all the words lie at leaf nodes, each internal node represents a phrase or a constituent of a sentence, and the root node represents the entire sentence [102, 99, 104]. In a dependency tree, each node including terminal and nonterminal nodes, represents a word, with dependency connections to other nodes [100, 45].

Figure 3.2 provides an example of a constituency-tree-based RvNN. As can be seen, the model first combine the word vectors of *historic* and *church* to form an intermediate representation h_1 to represents a noun phrase (np), which is then combined with x_1 to produce h_2 that represents the entire phrase *the historic church*. Specifically, the recursive formula is applied as the follows:

$$h_1 = f(W[x_2 : x_3] + b), \quad (\text{Eq. 3.12})$$

$$h_2 = f(W[x_1 : h_1] + b), \quad (\text{Eq. 3.13})$$

Figure 3.3: Attention network for machine translation [3].

where $[:]$ represents the concatenation of vectors, and W is a transformation matrix. The above example shows a simplest RvNN. To allow for more complicated modeling, we can use different transformation matrices that are tied within each syntactic relation. Moreover, the composition between any two child nodes could be replaced with tensor operations so that various correlations are considered.

3.1.3 Attention-based Models

Attentions [80] and memory networks [120] have recently been used for various machine learning tasks, including image generation [36], machine translation [3], sentence summarization [97], document sentiment classification [124], and question answering [40]. The attention mechanism aims to select and attend to relevant parts of the input which could be thought of as a soft-alignment process. This mechanism overcomes the limitation of RtNN for dealing with long-term dependencies by only focusing on relevant input units. A memory network generally consists of multiple layers of attentions, which has shown superior performance in many NLP tasks [55, 106].

The attention mechanism is inspired by the process of visual attentions when observing an image by a human. Ideally, one processes by focusing on some region of the

image with “high resolution”, but the other unimportant regions with “low resolution”. Extending to NLP tasks, the attention mechanism helps to select only the key components of a given text that are relevant to the task at hand. [3] first integrated the attention model into a RtNN for machine translation. Their proposed architecture is shown in Figure 3.3. Given a sentence written in a source language with a sequence of embeddings $\{x_1, x_2, \dots, x_T\}$, the required output is a sequence of words $\{y_1, y_2, \dots, y_N\}$ in the target language. Formally, the network involves an encoding operation to encode the source sequence. Here they used bidirectional RtNN to do the encoding such that $h_j = [\vec{h}_j^\top : \overleftarrow{h}_j^\top]^\top$, where $\vec{h}_j$ is the hidden state for a forward RtNN and $\overleftarrow{h}_j$ is the hidden state for a backward RtNN. Given $\{h_j\}$ ’s, the decoder will use attention mechanism to compute a vector representation c_i according to each h_j ’s relevance score:

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j, \quad (\text{Eq. 3.14})$$

with α_{tj} representing the relevance score for h_j obtained through

$$a_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}, \quad (\text{Eq. 3.15})$$

$$e_{tj} = g(s_{t-1}, h_j). \quad (\text{Eq. 3.16})$$

Here $g(s_{t-1}, h_j)$ is an alignment model to compute the correlation between the translation at time stamp $t - 1$ and the j th word in the source language. The attention vector c_t is then used to produce the hidden state s_t

$$s_t = f(s_{t-1}, y_{t-1}, c_t). \quad (\text{Eq. 3.17})$$

This construction allows for selecting different parts of the source sentence when translating to different positions in the target language.

Based on attention mechanism, memory networks were introduced by [120] for the problem of question answering. In this case, the memory serves as a dynamic knowledge

base that consists of four inference components: The component $\mathbf{I}$ is used to store the internal representations obtained by converting the input features. The component $\mathbf{G}$ is used to update the states of the memory. The component $\mathbf{O}$ is used to produce the output representations in the feature representation space. The component $\mathbf{R}$ is used to convert the output representation to the final response. [106] proposed an end-to-end memory network consisting of multiple layers of attention-based recurrent neural networks. This network could learn multiple levels of abstraction from the sentences for question answering. Each attention layer selects relevant part of a sentence according to the question and the output from the previous attention layer, and at the same time updates the memory states. The multi-layer construction helps to refine the learned knowledge for better predictions.

3.2 Problem Definition

This section provides a formal definition to the task of aspect and opinion terms extraction as well as the general notations that are used throughout this dissertation. Suppose that we are given a training set of customer review sentences in a specific domain, denoted by $S = \{s_1, \dots, s_N\}$, where N is the number of review sentences. For any $s_i \in S$, there may exist a set of aspect terms $A_i = \{a_{i1}, \dots, a_{il}\}$, where each $a_{ij} \in A_i$ can be a single word or a sequence of words expressing explicitly some aspect of an entity, and a set of opinion terms $O_i = \{o_{i1}, \dots, o_{im}\}$, where each o_{ir} can be a single word or a sequence of words expressing the subjective sentiment of the comment holder. The task is to learn a classifier to extract the set of aspect terms A_i and the set of opinion terms O_i from each review sentence $s_i \in S$.

This task can be formulated as a sequence tagging problem by using the BIO encoding scheme. Specifically, we denote a sentence by a sequence of tokens $s_i = \{w_{i1}, \dots, w_{in_i}\}$ and represent it as a $D \times n_i$ matrix $\mathbf{X}_i = \{\mathbf{x}_{i1}, \dots, \mathbf{x}_{in_i}\}$, where $\mathbf{x}_{ij} \in \mathbb{R}^D$ is a feature vector for

the j -th token of the sentence. Each word $w_{ip} \in s_i$ is labeled as one out of the following 5 classes: BA (beginning of aspect), IA (inside of aspect), BO (beginning of opinion), IO (inside of opinion), and N (none of the above). Let $\mathbf{L} = \{\text{BA}, \text{IA}, \text{BO}, \text{IO}, \text{N}\}$. We are also given a test set of review sentences denoted by $S' = \{s'_1, \dots, s'_{N'}\}$, where N' is the number of test reviews. For each test review $s'_i \in S'$, our objective is to predict the class label $y'_{iq} \in \mathbf{L}$ for each word w'_{iq} . Note that a sequence of predictions with BA at the beginning followed by IA are indication of one aspect, which is similar for opinion terms. Here, only exact match between predicted aspect/opinion terms and the ground-truth aspect/opinion terms is counted as correct.

Notation: Throughout this thesis, We use **bold letters** in their lowercase ($\mathbf{v}$) to represent vectors, **bold letters** in their UPPERCASE ($\mathbf{M}$) to represent matrices or tensors, and normal letters (a) to represent scalars. We use $[\mathbf{v} : \mathbf{r}]$ to denote the concatenation of vectors $\mathbf{v}$ and $\mathbf{r}$. Both $\odot$ and $\langle \cdot, \cdot \rangle$ denote element-wise multiplications. $\mathbf{v}_{[c]}$ represents the c -th element in vector $\mathbf{v}$. Similarly, $\mathbf{M}_{[r, \cdot]}$ and $\mathbf{M}_{[\cdot, c]}$ represent the r -th row and c -th column, respectively, in matrix $\mathbf{M}$. $f_{\text{GRU}}(\mathbf{x}, \mathbf{h}; \Theta)$ denotes the computation of a GRU network with input unit $\mathbf{x}$, hidden unit $\mathbf{h}$ and a parameter set Θ . For all the proposed models, we use $\mathbf{X}$ to denote the input matrix where each column $\mathbf{x}$ is a pre-trained word embedding. $\mathbf{H}$ denotes the hidden matrix where each column $\mathbf{h}$ is the hidden vector for a word in the input sequence.

Chapter 4

Exploiting Token-level Interactions via Dependency Trees

This chapter introduces a novel joint model combining graphical models and deep learning models for explicit aspect terms and opinion terms extraction. Among previous work for aspect/opinion terms extraction, one of the approaches is to accumulate aspect and opinion terms from a seed collection without label information, by utilizing syntactic rules or modification relations between them [94, 67]. For example, in a restaurant review “*I have to say they have one of the fastest delivery times in the city.*”, the aspect term is *delivery times*, and the opinion term is *fastest*. If we know *fastest* is an opinion word, then *delivery times* is probably inferred to be an aspect because *fastest* is its modifier. However, this approach largely relies on hand-coded rules and is restricted to certain Part-of-Speech (POS) tags, e.g., opinion words are restricted to be adjectives. Another approach focuses on feature engineering based on predefined lexicons, syntactic analysis, etc. [48, 61]. A sequence labeling classifier is then built to extract aspect and opinion terms. This approach requires extensive efforts for designing hand-crafted features and only combines features linearly for classification which ignores higher order interactions.

To overcome the limitations of existing methods, we propose a novel model, named Recursive Neural Conditional Random Fields (RNCRF). Specifically, RNCRF consists of two main components. The first component is a recursive neural network (RvNN) [102]

based on a dependency tree of each sentence. The goal is to learn a high-level feature representation for each word in the context of each sentence and make the representation learning for aspect and opinion terms interactive through the underlying dependency structure among them. The output of the RvNN is then fed into a Conditional Random Field (CRF) [56] to learn a discriminative mapping from high-level features to labels, i.e., *aspects*, *opinions*, or *others*, so that context information can be well captured. Our main contributions are to use RvNN for encoding aspect-opinion relations in high-level representation learning and to present a joint optimization approach based on maximum likelihood and backpropagation to learn the RvNN and CRF components simultaneously. In this way, the label information of aspect and opinion terms can be dually propagated from parameter learning in CRF to representation learning in RvNN. We conduct expensive experiments on the dataset from SemEval challenge 2014 task 4 (subtask 1) [90] to verify the superiority of RNCRF over several baseline methods as well as the winning systems of the challenge.

4.1 Motivation

Previous studies have shown that exploiting connections between aspect and opinion terms is promising for the extraction task, because some common patterns exist between aspect terms and their corresponding opinion terms. Dependency trees, on the other hand, provide syntactic structures of a given sentence that consist of dependency relations among related tokens. Given a parsed tree, a RvNN could be built that computes a high-level representation for each word considering the information of its dependents. In this way, the common patterns between aspect terms and opinion terms could be learned automatically through label propagations. Besides the syntactic structure, sequential correlations are also crucial to model how the context window affects the prediction of any given word. This can be achieved through graphical models, e.g., CRFs. Traditional

Figure 4.1: A linear-chain CRF.

CRFs take discrete features as the input with linear combinations. In order to exploit more complex interactions, we feed the CRF with the hidden representations of RvNN. We propose a novel joint model that integrates RvNN and CRFs into a unified framework to simultaneously model the syntactic and sequential correlations among different tokens.

4.2 Introduction to Conditional Random Fields

Conditional Random Fields (CRFs) are discriminative graphical models for predictions over structured data, e.g., sequences or trees. In the case of sequential predictions, we apply a linear-chain CRF to learn the sequential interactions within the context of each word in a sentence.

Formally, given a sentence, the entire structure can be modeled as an undirected graph shown in Figure 4.1. Given the input sequence $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, the model computes a sequence of output labels $\mathbf{y} = \{y_1, y_2, \dots, y_n\} \in \mathbf{Y}$, where $\mathbf{Y}$ is the collection of all possible combinations of different classes. The CRF computes a score that represents the probability of a label sequence given its input:

$$p(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \Psi_c(\mathbf{x}, \mathbf{y}_c), \quad (\text{Eq. 4.1})$$

where $C = \{U, P\}$ represents the set of different cliques in the graph. Here U indicates an unary clique connecting input to the output and P indicates a pairwise clique connecting adjacent output labels. Here Ψ_c is defined as:

$$\Psi_c(\mathbf{x}, \mathbf{y}_c) = \exp \langle W_c, g(\mathbf{x}, \mathbf{y}_c) \rangle, \quad (\text{Eq. 4.2})$$

Figure 4.2: Examples of a dependency tree and its corresponding DT-RvNN structure for a review sentence.

which computes the potential of a given clique c , with $g(\cdot, \cdot)$ generating a feature vector of the input. $\mathbf{W}_c$ is tied for each different $\mathbf{y}_c$, which indicates the labels for clique c . The operator $\langle \cdot, \cdot \rangle$ is the element-wise multiplication, and $Z(\mathbf{x})$ is a normalization term that runs over all possible output combinations. At inference, the model will output $\hat{\mathbf{y}}$ with the maximum conditional probability $p(\hat{\mathbf{y}} | \mathbf{x})$.

4.3 Recursive Neural Conditional Random Fields

As described in the beginning of this chapter, RNCRF consists of two main components: 1) a dependency-tree-based recursive neural network (DT-RvNN) to learn a high-level representation for each word in a sentence, and 2) a CRF to take the learned representation as input to capture context around each word for explicit aspect and opinion terms extraction. Next, we present these two components in details.

4.3.1 Dependency-tree Recursive Neural Network

We begin by associating each word w in our vocabulary with a feature vector $\mathbf{x} \in \mathbb{R}^D$, which corresponds to a column of a word embedding matrix $\mathbf{W}_e \in \mathbb{R}^{D \times v}$, where v is the size of the vocabulary. For each sentence, we build a DT-RvNN based on the corresponding dependency parse tree with word embeddings as initialization. An example of the

dependency parse tree is shown in Figure 4.2.a, where each edge starts from the parent and points to its dependent with a syntactic relation.

In a DT-RvNN, each node n , including leaf nodes, internal nodes and the root node, in a specific sentence is associated with a word w , an input feature vector $\mathbf{x}_w$, and a hidden vector $\mathbf{h}_n \in \mathbb{R}^d$. Each dependency relation r is associated with a separate matrix $\mathbf{W}_r \in \mathbb{R}^{d \times d}$. In addition, a common transformation matrix $\mathbf{W}_v \in \mathbb{R}^{d \times D}$ is introduced to map the word embedding $\mathbf{x}_w$ at node n to its corresponding hidden vector $\mathbf{h}_n$.

Along with a particular dependency tree, a hidden vector $\mathbf{h}_n$ is computed from its own word embedding $\mathbf{x}_w$ at node n with the transformation matrix $\mathbf{W}_v$ and its children's hidden vectors $\mathbf{h}_{\text{child}(n)}$ with the corresponding relation matrices $\{\mathbf{W}_r\}$'s. For instance, given the parse tree shown in Figure 4.2.a, we first compute the leaf nodes associated with the words *I* and *the* using $\mathbf{W}_v$ as follows,

$$\begin{aligned}\mathbf{h}_I &= f(\mathbf{W}_v \cdot \mathbf{x}_I + \mathbf{b}), \\ \mathbf{h}_{\text{the}} &= f(\mathbf{W}_v \cdot \mathbf{x}_{\text{the}} + \mathbf{b}),\end{aligned}$$

where f is a nonlinear activation function and $\mathbf{b}$ is a bias term. In this paper, we adopt $\tanh(\cdot)$ as the activation function. Once the hidden vectors of all the leaf nodes are generated, we can recursively generate hidden vectors for interior nodes using the corresponding relation matrix $\mathbf{W}_r$ and the common transformation matrix $\mathbf{W}_v$ as follows,

$$\begin{aligned}\mathbf{h}_{\text{food}} &= f(\mathbf{W}_v \cdot \mathbf{x}_{\text{food}} + \mathbf{W}_{\text{DET}} \cdot \mathbf{h}_{\text{the}} + \mathbf{b}), \\ \mathbf{h}_{\text{like}} &= f(\mathbf{W}_v \cdot \mathbf{x}_{\text{like}} + \mathbf{W}_{\text{DOBJ}} \cdot \mathbf{h}_{\text{food}} \\ &\quad + \mathbf{W}_{\text{NSUBJ}} \cdot \mathbf{h}_I + \mathbf{b}).\end{aligned}$$

The resulting DT-RvNN is shown in Figure 4.2.b. In general, a hidden vector for any node n associated with a word vector $\mathbf{x}_w$ can be computed as follows,

$$\mathbf{h}_n = f \left(\mathbf{W}_v \cdot \mathbf{x}_w + \mathbf{b} + \sum_{\mathbf{k} \in \mathcal{K}_n} \mathbf{W}_{\mathbf{r}_{\mathbf{n}\mathbf{k}}} \cdot \mathbf{h}_{\mathbf{k}} \right), \quad (\text{Eq. 4.3})$$

Figure 4.3: RNCRF structure for a review sentence.

where $\mathcal{K}_n$ denotes the set of children of node n , r_{nk} denotes the dependency relation between node n and its child node k , and $\mathbf{h}_k$ is the hidden vector of the child node k . The parameters of DT-RvNN, $\Theta_{\text{RvNN}} = \{\mathbf{W}_v, \mathbf{W}_r, \mathbf{W}_e, \mathbf{b}\}$, are learned during training.

4.3.2 Integration with CRFs

In RNCRF, we feed the output of DT-RvNN, i.e., the hidden representation of each word in a sentence, to a CRF. Updates of parameters for RNCRF are carried out successively from the top to bottom, by propagating errors through CRF to the hidden layers of RvNN (including word embeddings) using backpropagation through structure (BPTS) [33].

Formally, for each sentence s_i , we denote the input for CRF by $\mathbf{H}_i$, which is generated by DT-RvNN. Here $\mathbf{H}_i$ is a matrix with columns of hidden vectors $\{\mathbf{h}_{i1}, \dots, \mathbf{h}_{in_i}\}$ to represent a sequence of words $\{w_{i1}, \dots, w_{in_i}\}$ in a sentence s_i . The model computes a structured output $\mathbf{y}_i = \{y_{i1}, \dots, y_{in_i}\} \in \mathbf{Y}$, where $\mathbf{Y}$ is a set of possible combinations of labels in label set $\mathbf{L}$. The entire structure can be represented by an undirected graph $G = (V, E)$ with cliques $c \in C$. In this paper, we employed linear-chain CRF, which has two different cliques: unary cliques (U) representing input-output connection, and pairwise cliques (P) representing adjacent output connections, as shown in Figure 4.3.

Figure 4.4: An example for computing input-output potential for the second position *like*.

During inference, the model aims to output $\hat{\mathbf{y}}$ with the maximum conditional probability $p(\mathbf{y}|\mathbf{H})$. (We drop the subscript i here for simplicity.) The probability is computed from potential outputs of the cliques:

$$p(\mathbf{y}|\mathbf{H}) = \frac{1}{Z(\mathbf{H})} \prod_{c \in C} \psi_c(\mathbf{H}, \mathbf{y}_c), \quad (\text{Eq. 4.4})$$

where $Z(\mathbf{H})$ is the normalization term, and $\psi_c(\mathbf{H}, \mathbf{y}_c)$ is the potential of clique c , computed as $\psi_c(\mathbf{H}, \mathbf{y}_c) = \exp \langle \mathbf{W}_c, F(\mathbf{H}, \mathbf{y}_c) \rangle$, where the RHS is the exponential of a linear combination of feature vector $F(\mathbf{H}, \mathbf{y}_c)$ for clique c , and the weight vector $\mathbf{W}_c$ is tied for unary and pairwise cliques. We also incorporate a context window of size $2T+1$ when computing unary potentials (T is a hyper-parameter). Thus, the potential of unary clique at node k can be written as

$$\begin{aligned} \psi_U(\mathbf{H}, y_k) = \exp & \left((\mathbf{W}_0)_{y_k} \cdot \mathbf{h}_k + \sum_{t=1}^T (\mathbf{W}_{-t})_{y_k} \cdot \mathbf{h}_{k-t} \right. \\ & \left. + \sum_{t=1}^T (\mathbf{W}_{+t})_{y_k} \cdot \mathbf{h}_{k+t} \right), \end{aligned} \quad (\text{Eq. 4.5})$$

where $\mathbf{W}_0$, $\mathbf{W}_{+t}$ and $\mathbf{W}_{-t}$ are weight matrices of the CRF for the current position, the t -th position to the right, and the t -th position to the left within context window, respectively. The subscript y_k indicates the corresponding row in the weight matrix.

For instance, Figure 4.4 shows an example of window size 3. At the second position, the input features for *like* are composed of the hidden vectors at position 1 ($\mathbf{h}_I$), position 2 ($\mathbf{h}_{\text{like}}$) and position 3 ($\mathbf{h}_{\text{the}}$). Therefore, the conditional distribution for the entire

sequence $\mathbf{y}$ in Figure 4.3 can be calculated as

$$p(\mathbf{y}|\mathbf{H}) = \frac{1}{Z(\mathbf{H})} \exp \left(\sum_{k=1}^4 (\mathbf{W}_0)_{y_k} \cdot \mathbf{h}_k + \sum_{k=2}^4 (\mathbf{W}_{-1})_{y_k} \cdot \mathbf{h}_{k-1} \right. \\ \left. + \sum_{k=1}^3 (\mathbf{W}_{+1})_{y_k} \cdot \mathbf{h}_{k+1} + \sum_{k=1}^3 \mathbf{V}_{y_k, y_{k+1}} \right),$$

where the first three terms in the exponential of the RHS consider unary clique while the last term considers the pairwise clique with matrix $\mathbf{V}$ representing pairwise state transition score. For simplicity in description on parameter updates, we denote the log-potential for clique $c \in \{U, P\}$ by $g_c(\mathbf{H}, \mathbf{y}_c) = \langle \mathbf{W}_c, F(\mathbf{H}, \mathbf{y}_c) \rangle$.

4.3.3 Joint Training for RNCRF

Through the objective of maximum likelihood, updates for parameters of RNCRF are first conducted on the parameters of the CRF (unary weight matrices $\Theta_U = \{\mathbf{W}_0, \mathbf{W}_{+t}, \mathbf{W}_{-t}\}$ and pairwise weight matrix $\mathbf{V}$) by applying chain rule to log-potential updates. Below is the gradient for Θ_U (updates for $\mathbf{V}$ are similar through the log-potential of pairwise clique $g_P(y'_k, y'_{k+1})$):

$$\Delta \Theta_U = \frac{\partial -\log p(\mathbf{y}|\mathbf{H})}{\partial g_U(\mathbf{H}, y'_k)} \cdot \frac{\partial g_U(\mathbf{H}, y'_k)}{\partial \Theta_U}, \quad (\text{Eq. 4.6})$$

where

$$\frac{\partial -\log p(\mathbf{y}|\mathbf{H})}{\partial g_U(\mathbf{H}, y'_k)} = -(1_{y_k=y'_k} - p(y'_k|\mathbf{H})), \quad (\text{Eq. 4.7})$$

and y'_k represents possible label configuration of node k . The hidden representations of each word and the parameters of DT-RvNN are updated subsequently by applying chain

Algorithm 1: Recursive Neural CRFs

Input: A set of customer review sequences: $S = \{s_1, \dots, s_N\}$, and feature vectors of D dimensions for each word $\{\mathbf{x}_w\}$'s, window size T for CRFs

Output: Parameters: $\Theta = \{\Theta_{\text{RvNN}}, \Theta_U, V\}$

Initialization: Initialize $\mathbf{W}_e$ using *word2vec*. Initialize $\mathbf{W}_v$ and $\{\mathbf{W}_r\}$'s randomly with uniform distribution between $\left[-\frac{\sqrt{6}}{\sqrt{2d+1}}, \frac{\sqrt{6}}{\sqrt{2d+1}}\right]$. Initialize $\mathbf{W}_0$, $\{\mathbf{W}_{+t}\}$'s, $\{\mathbf{W}_{-t}\}$'s, $\mathbf{V}$, and $\mathbf{b}$ with all 0's
for each sentence s_i **do**

1: Use DT-RvNN (Eq. 4.3) to generate $\mathbf{H}_i$

2: Compute $p(\mathbf{y}_i|\mathbf{H}_i)$ using (Eq. 4.4)

3: Use the backpropagation algorithm to update parameters Θ through (Eq. 4.6)-(Eq. 4.10)

end for

rule with (Eq. 4.7) through BPTS as follows,

$$\Delta \mathbf{h}_{\text{root}} = \frac{\partial -\log p(\mathbf{y}|\mathbf{H})}{\partial g_U(\mathbf{H}, y'_{\text{root}})} \cdot \frac{\partial g_U(\mathbf{H}, y'_{\text{root}})}{\partial \mathbf{h}_{\text{root}}}, \quad (\text{Eq. 4.8})$$

$$\begin{aligned} \Delta \mathbf{h}_{k \neq \text{root}} &= \frac{\partial -\log p(\mathbf{y}|\mathbf{H})}{\partial g_U(\mathbf{H}, y'_k)} \cdot \frac{\partial g_U(\mathbf{H}, y'_k)}{\partial \mathbf{h}_k} \\ &\quad + \Delta \mathbf{h}_{\text{par}(k)} \cdot \frac{\partial \mathbf{h}_{\text{par}(k)}}{\partial \mathbf{h}_k}, \end{aligned} \quad (\text{Eq. 4.9})$$

$$\Delta \Theta_{\text{RvNN}} = \sum_{k=1}^K \frac{\partial -\log p(\mathbf{y}|\mathbf{H})}{\partial \mathbf{h}_k} \cdot \frac{\partial \mathbf{h}_k}{\partial \Theta_{\text{RvNN}}}, \quad (\text{Eq. 4.10})$$

where $\mathbf{h}_{\text{root}}$ represents the hidden vector of the word pointed by ROOT in the corresponding DT-RvNN. Since this word is the topmost node in the tree, it only inherits error from the CRF output. In (Eq. 4.9), $\mathbf{h}_{\text{par}(k)}$ denotes the hidden vector of the parent node of node k in DT-RvNN. Hence the lower nodes receive error from both the CRF output and error propagation from parent node. The parameters within DT-RvNN, Θ_{RvNN} , are updated by applying chain rule with respect to updates of hidden vectors, and aggregating among all associated nodes, as shown in (Eq. 4.10). The overall procedure of RNCRF is summarized in Algorithm 1.

4.3.4 Advantage of RNCRF

The best performing system [113] for SemEval challenge 2014 task 4 (subtask 1) employed CRFs with extensive hand-crafted features including those induced from dependency

trees. However, their experiments showed that the addition of the features induced from dependency relations does not improve the performance. This indicates the difficulty of incorporating dependency structure explicitly as input features, which motivates the design of our model to use DT-RvNN to encode dependency between words for feature learning. The most important advantage of RNCRF is the ability to learn the underlying dual propagation between aspect and opinion terms from the tree structure itself. Specifically as shown in Figure 4.3, where the aspect is *food* and the opinion expression is *like*. In the dependency tree, *food* depends on *like* with the relation *DOBJ*. During training, RNCRF computes the hidden vector $\mathbf{h}_{\text{like}}$ for *like*, which is also obtained from $\mathbf{h}_{\text{food}}$. As a result, the prediction for *like* is affected by $\mathbf{h}_{\text{food}}$. This is one-way propagation from *food* to *like*. During backpropagation, the error for *like* is propagated through a top-down manner to revise the representation $\mathbf{h}_{\text{food}}$. This is the other-way propagation from *like* to *food*. Therefore, the dependency structure together with the learning approach help to enforce the dual propagation of aspect-opinion pairs as long as the dependency relation exists, either directly or indirectly.

Adding Linguistic/Lexicon Features: RNCRF is an end-to-end model, where feature engineering is not necessary. However, it is flexible to incorporate light hand-crafted features into RNCRF to further boost its performance, such as features from POS tags, name-list, or sentiment lexicon. These features could be appended to the hidden vector of each word, but keep fixed during training, unlike learnable neural inputs and the CRF weights as described in Section 4.3.3. As will be shown in experiments, RNCRF without any hand-crafted features slightly outperforms the best performing systems that involve heavy feature engineering efforts, and RNCRF with light feature engineering can achieve even better performance.

Domain	Training	Test	Total
Restaurant	3,041	800	3,841
Laptop	3,045	800	3,845
Total	6,086	1,600	7,686

Table 4.1: SemEval Challenge 2014 task 4 dataset

4.4 Experiment

4.4.1 Dataset & Experimental Setup

We evaluate our model on the dataset from SemEval Challenge 2014 task 4 (subtask 1), which includes reviews from two domains: restaurant and laptop. The detailed description of the dataset is given in Table 4.1. As the original dataset only includes manually annotated labels for aspect terms but not for opinion terms, we manually annotated opinion terms for each sentence by ourselves to facilitate our experiments.

For word vector initialization, we train word embeddings with word2vec [76] on the Yelp Challenge dataset¹ for the restaurant domain and on the Amazon reviews dataset² [73] for the laptop domain. The Yelp dataset contains 2.2M restaurant reviews with 54K vocabulary size. For the Amazon reviews, we only extracted the electronic domain that contains 1M reviews with 590K vocabulary size. We vary different dimensions for word embeddings and chose 300 for both domains. Empirical sensitivity studies on different dimensions of word embeddings are also conducted. Dependency trees are generated using Stanford Dependency Parser [52]. Regarding CRFs, we implement a linear-chain CRF using CRFSuite [84]. Because of the relatively small size of training data and a large number of parameters, we perform pretraining on the parameters of DT-RvNN with cross-entropy error, which is a common strategy for deep learning [30]. We implement mini-batch stochastic gradient descent (SGD) with a batch size of 25,

¹http://www.yelp.com/dataset_challenge

²<http://jmcauley.ucsd.edu/data/amazon/links.html>

and an adaptive learning rate (AdaGrad) initialized at 0.02 for pretraining of DT-RvNN, which runs 4 epochs for the restaurant domain and 5 epochs for the laptop domain. For parameter learning of the joint model RNCRF, we implement SGD with a decaying learning rate initialized at 0.02. We also try with varying context window size, and use 3 for the laptop domain and 5 for the restaurant domain, respectively. All parameters are chosen by cross validation.

As discussed in Section 4.3.4, hand-crafted features can be easily incorporated into RNCRF. We generate three types of simple features based on POS tags, name-list and sentiment lexicon to show further improvement by incorporating these features. Following [113], we extract two sets of name list from the training data for each domain, where one includes high-frequency aspect terms, and the other includes high-probability aspect words. These two sets are used to construct two lexicon features, i.e. we build a 2D binary vector: if a word is in a set, the corresponding value is 1, otherwise 0. For POS tags, we use Stanford POS tagger [114], and convert them to universal POS tags that have 15 different categories. We then generate 15 one-hot POS tag features. For sentiment lexicon, we use the collection of commonly used opinion words (around 6,800) [42]. Similar to name list, we create a binary feature to indicate whether the word belongs to opinion lexicon. We denote by RNCRF+F the proposed model with the three types of features.

Compared to the winning systems of SemEval Challenge 2014 task 4 (subtask 1), RNCRF or RNCRF+F uses additional labels of opinion terms for training. Therefore, to conduct fair comparison experiments with the winning systems, we implement RNCRF-O by omitting opinion labels to train our model (i.e., labels become “BA”, “IA”, “N”). Accordingly, we denote by RNCRF-O+F the RNCRF-O model with the three additional types of hand-crafted features.

4.4.2 Result & Analysis

We compare our model with several baselines:

- **CRF-1**: a linear-chain CRF with standard linguistic features including word string, stylistics, POS tag, context string, and context POS tags.
- **CRF-2**: a linear-chain CRF with both standard linguistic features and dependency information including head word, dependency relations with parent token and child tokens.
- **LSTM**: an LSTM network built on top of word embeddings proposed by [66]. We keep original settings in [66] but replace their word embeddings with ours (300 dimension). We try different hidden layer dimensions (50, 100, 150, 200) and reported the best result with size 50.
- **LSTM+F**: the above LSTM model with the three additional types of hand-crafted features as with RNCRF.
- **SemEval-1, SemEval-2**: the top two winning systems for SemEval challenge 2014 task 4 (subtask 1).
- **WDEmb+B+CRF**³: the model proposed by [125] using word and dependency path embeddings combined with linear context embedding features, dependency context embedding features and hand-crafted features (i.e., feature engineering) as CRF input.

The comparison results are shown in Table 4.2 for both the restaurant domain and the laptop domain. Note that we provide the same annotated dataset (both aspect labels and opinion labels are included for training) for CRF-1, CRF-2 and LSTM for fair comparison.

³We report the best results from the original paper [125].

	Restaurant		Laptop	
Models	Aspect	Opinion	Aspect	Opinion
SemEval-1	84.01	-	74.55	-
SemEval-2	83.98	-	73.78	-
WDEmb+B+CRF	84.97	-	75.16	-
CRF-1	77.00	78.95	66.21	71.78
CRF-2	78.37	78.65	68.35	70.05
LSTM	81.15	80.22	72.73	74.98
LSTM+F	82.99	82.90	73.23	77.67
RNCRF-O	82.73	-	74.52	-
RNCRF-O+F	84.25	-	77.26	-
RNCRF	84.05	80.93	76.83	76.76
RNCRF+F	84.93	84.11	78.42	79.44

Table 4.2: Comparison results in terms of F1 scores.

It is clear that our proposed model RNCRF achieves superior performance compared with most of the baseline models. The performance is even better by adding simple hand-crafted features, i.e., RNCRF+F, with 0.92% and 3.87% absolute improvement over the best system in the challenge for aspect extraction for the restaurant domain and the laptop domain, respectively. This shows the advantage of combining high-level continuous features and discrete hand-crafted features. Though CRFs usually show promising results in sequence tagging problems, they fail to achieve comparable performance when lacking of extensive features (e.g., CRF-1). By adding dependency information explicitly in CRF-2, the result only improves slightly for aspect extraction. Alternatively, by incorporating dependency information into a deep learning model (e.g., RNCRF), the result shows more than 7% improvement for aspect extraction and 2% for opinion extraction.

By removing the labels for opinion terms, RNCRF-O produces inferior results than RNCRF because the effect of dual propagation of aspect and opinion pairs disappears with the absence of opinion labels. This verifies our previous assumption that DT-RvNN could learn the interactive effects within aspects and opinions. However, the performance of RNCRF-O is still comparable to the top systems and even better with the addition

	Restaurant		Laptop	
Models	Aspect	Opinion	Aspect	Opinion
DT-RvNN + SoftMax	72.45	69.76	66.11	64.66
CRF + word2vec	82.57	78.83	63.62	56.96
RNCRF	84.05	80.93	76.83	76.76
RNCRF+POS	84.08	81.48	77.04	77.45
RNCRF+NL	84.24	81.22	78.12	77.20
RNCRF+Lex	84.21	84.14	77.15	78.56
RNCRF+F	84.93	84.11	78.42	79.44

Table 4.3: Impact of different components.

of simple linguistic features: 0.24% and 2.71% superior than the best system in the challenge for the restaurant domain and the laptop domain, respectively. This shows the robustness of our model even without additional opinion labels.

LSTM has shown comparable results for aspect extraction [66]. However, in their work, they used well-pretrained word embeddings by training with large corpus or extensive external resources, e.g., chunking, and NER. To compare their model with RNCRF, we re-implement LSTM with the same word embedding strategy and labeling resources as ours. The results show that our model outperforms LSTM in aspect extraction by 2.90% and 4.10% for the restaurant domain and the laptop domain, respectively. We conclude that a standard LSTM model fails to extract the relations between aspect and opinion terms. Even with the addition of same linguistic features, LSTM is still inferior than RNCRF itself in terms of aspect extraction. Moreover, our result is comparable with WDEmb+B+CRF in the restaurant domain and better in the laptop domain (+3.26%). Note that WDEmb+B+CRF appended dependency context information into CRF while our model encode such information into high-level representation learning.

To test the impact of each component of RNCRF and the three types of hand-crafted features, we conduct experiments on different model settings:

4.5.a: On the restaurant domain.

4.5.b: On the laptop domain.

Figure 4.5: Sensitivity studies on word embeddings.

- **DT-RvNN+SoftMax**: rather than using a CRF, a softmax classifier is used on top of DT-RvNN.
- **CRF+word2vec**: a linear-chain CRF with word embeddings only without using DT-RvNN.
- **RNCRF+POS/NL/Lex**: the RNCRF model with POS tag or name list or sentiment lexicon feature(s).

The comparison results are shown in Table 4.3. Similarly, both aspect and opinion term labels are provided for training for each of the above models. Firstly, RNCRF achieves much better results compared to DT-RvNN+SoftMax (+11.60% and +10.72% for the restaurant domain and the laptop domain in aspect extraction). This is because DT-RvNN fails to fully exploit context information for sequence labeling, which, in contrast, can be achieved by CRF. Secondly, RNCRF outperforms CRF+word2vec, which proves the importance of DT-RvNN for modeling interactions between aspects and opinions. Hence, the combination of DT-RvNN and CRF inherits the advantages from both models.

Moreover, by separately adding hand-crafted features, we can observe that name-list-based features and the sentiment lexicon feature are most effective for aspect extraction and opinion extraction, respectively. This may be explained by the fact that name-list-based features usually contain informative evidence for aspect terms and sentiment lexicon provides explicit indication about opinions.

Besides the comparison experiments, we also conduct sensitivity test for our proposed model in terms of word vector dimensions. We tested a set of different dimensions ranging from 25 to 400, with 25 increment. The sensitivity plot is shown in Figure 4.5. The performance for aspect extraction is smooth with different vector lengths for both domains. For restaurant domain, the result is stable when dimension is larger than or equal to 100, with the highest at 325. For the laptop domain, the best result is at dimension 300, but with relatively small variations. For opinion extraction, the performance reaches a good level when the dimension is larger than or equal to 75 for the restaurant domain and 125 for the laptop domain. This proves the stability and robustness of our model.

4.5 Summary

We have presented a joint model, RNCRF, that achieves the state-of-the-art performance for explicit aspect and opinion term extraction on a benchmark dataset. With the help of DT-RvNN, high-level features can be learned by encoding the underlying dual propagation of aspect-opinion pairs. RNCRF combines the advantages of DT-RvNNs and CRFs, and thus outperforms the traditional rule-based methods in terms of flexibility, because aspect terms and opinion terms are not only restricted to certain observed relations and POS tags. Compared to feature engineering methods with CRFs, the proposed model saves much effort in composing features, and it is able to extract higher-level features obtained from non-linear transformations.

Chapter 5

Modeling Token-level Interactions via Attentions

This chapter describes an end-to-end attention-based model for explicit aspect terms and opinion terms extraction. Compared with Chapter 4, this proposed model is able to learn the complex correlations among the tokens automatically without any pre-generated parsed trees. The method is further extended to simultaneously extract the target terms and categorize them into a set of pre-defined aspect categories, providing more complete information for structured opinion output.

For aspect and opinion terms extraction, deep-learning-based approaches [66, 125] have been proposed to learn high-level representations for each token, on which a classifier can be trained. Chapter 4 applies dependency-tree-based recursive neural network to dually propagate the information between aspect terms and opinion terms. Despite some promising results, most of these approaches require a parser analyzing the syntactic/dependency structure of the sentence to be encoded into the deep models. Therefore, the performances of these approaches rely on the quality of the parsing results. In practice, the dependency structures of many user-generated texts may not be precise with a computational parser, especially in informal texts, which may degrade the performances of existing approaches. Therefore, we propose to use the attention mechanism [3] with

tensor operators in a memory network to replace the role of dependency parsers to automatically capture the relations among tokens in a sentence. Specifically, we design a couple of attentions, one for aspects extraction and the other for opinions extraction. They are learned interactively such that label information can be dually propagated among aspect terms and opinion terms by exploiting their relations. Moreover, we use a memory network to explore multiple layers of the coupled attentions in order to extract inconspicuous aspect/opinion terms.

Going further, we extend the extraction task to a finer-grained problem named *category-specific aspect and opinion terms extraction*, where aspect/opinion terms need to be extracted and classified to a category from a pre-defined set, simultaneously. This could provide a more structured opinion outputs and is also beneficial for linking aspect terms and opinion terms through their category information. For example, in the sentence “*The soup is served with nice portion, the service is prompt*”, the objective is to extract and classify *soup* and *portion* as aspect terms under the “DRINKS” category, and *service* as an aspect term under the “SERVICE” category, similar for the opinion terms *nice* and *prompt*.

To this extent, some previous works focus on categorization of aspect terms, where aspect terms are extracted in advance, and the goal is to classify them into one of the predefined categories [10, 128, 130, 131]. The joint task is much more challenging and has rarely been investigated because when specific categories are taken into consideration for terms extraction, training data may become extremely sparse, e.g. certain categories may only contain very few reviews or sentences. Moreover, it requires to achieve both extraction and categorization, simultaneously, which significantly increases the difficulty compared with the task of only extracting overall aspect/opinion terms or classifying pre-extracted terms. Although topic models [37, 111] can achieve both grouping and extraction at the same time, they mainly focus on grouping, and could only identify

general and coarse-grained aspect terms. To solve the problem, we offer an end-to-end deep multi-task learning architecture. Our high-level idea is that we consider terms extraction for each specific category as an individual task, where we can use the proposed memory network introduced in the first stage for co-extracting aspect and opinion terms. The memory networks are then jointly learned in a multi-task learning network to address the data sparsity issue of each task.

In summary, our contributions are 3-fold: 1) We propose an end-to-end memory network for aspect and opinion terms co-extraction without requiring any syntactic/dependency parsers or linguistic resources to generate additional information as input. 2) We introduce a finer-grained problem and extend the memory network with a multi-task mechanism to solve it. 3) We conduct extensive experiments on SemEval Challenge benchmark datasets to demonstrate state-of-the-art performance of our proposed approaches.

5.1 Multi-task Learning

Multi-task learning aims to improve generalization for each individual task by exploiting relatedness among different tasks [11]. One common assumption in multi-task learning is that parameters for different tasks lie in a low-dimensional subspace [2, 54] which is achieved either by imposing low-rank constraint or matrix factorization. Through factorization, the model of each task becomes a linear combination of a small set of latent tasks. Following this idea, a multi-linear model was proposed in [95] to deal with multi-modal tasks with multiple indexes. This tensor factorization idea also promotes a deep multi-task learning model [124] where the parameters in different layers of a CNN for different tasks form a tensor that could be factorized across tasks. Moreover, many deep learning models have been introduced for multi-task learning [68, 79] with an aim to learn shared hidden representation that are regularized from different tasks. Our proposed deep multi-task learning model is specially designed for sentiment analysis and

is expected to be more effective for the proposed finer-grained sentiment analysis problem compared with other general methods.

5.2 Problem Definition & Motivation

Problem Definition: Besides all the notations introduced in Section 3.2, this section provides some additional definitions related to categories to facilitate illustration in the subsequent sections. Considering category information, let $\mathcal{C} = \{1, 2, \dots, C\}$ denote a predefined set of C categories, where $c \in \mathcal{C}$ is an entity/attribute type, e.g., “DRINK#QUALITY” in the restaurant domain. A superscript c denotes the category-related variable. We denote $\mathbf{y}_i^c \in \mathbb{R}^{n_i}$, where $y_{ij}^c \in \{\text{BA}_c, \text{IA}_c, \text{BP}_c, \text{IP}_c, \text{N}_c\}$ is the label of the j -th token. Here, BA_c and IA_c refer to *beginning of aspect* and *inside of aspect*, respectively, of **category** c , similar for BP_c , IP_c and N_c . In the following, we use j to denote the index of a token in a sentence, c to denote the association with category c and for simplifying notations, we omit the sentence index i if the context is clear.

Motivation: To fully exploit the syntactic relations among different tokens in a sentence, most existing methods applied a computational parser to analyze the syntactic/dependency structure of each sentence in advance and use the relations between aspects and opinions to double propagate the information. One major limitation is that the generated relations are deterministic and fail to handle uncertainty underlying the data. It is even worse when grammar and syntactic errors commonly exist in user-generated texts, in which case the outputs of a dependency parser may not be precise, and thus degrades the performance. To avoid this, we propose to use memory networks with coupled attentions (MNCA) to automatically learn the relations between aspect terms and opinion terms without any linguistic knowledge.

To further explore category information for each aspect/opinion terms, one straightforward solution is to apply the extraction model to identify general aspect/opinion terms

Figure 5.1: A dependency example and illustration for the features of MNCA.

first, and then post-classify them into different categories using an additional classifier. However, this pipeline approach may suffer from error propagation from the extraction phase to the classification phase. An alternative solution is to train an extraction model for each category c independently, and then combine the results of all the extraction models to generate final predictions. However, for each fine-grained category, aspect and opinion terms become extremely sparse for training, which makes it difficult to learn a precise model for each category if trained independently. To address the above issues, we propose to model the problem in a multi-task learning manner, where aspect/opinion terms extraction for each category is considered as an individual task, and an end-to-end deep learning architecture is developed to jointly learn the tasks by exploiting their commonalities and similarities. We name our proposed multi-task model as Multi-task Memory Networks (MTMNs). In the following section, we first introduce our memory network with coupled attentions, namely MNCA, for target terms extraction which is the basic component of MTMN. Then we present the detailed architecture for MTMN for category-specific aspect and opinion terms co-extraction.

5.3 Memory Network with Coupled Attentions for Extraction

In order to extract the interactions between aspect terms and opinion terms automatically, we propose a novel memory network which consists of the following features:

- For each sentence, we construct a pair of attentions: an aspect attention for aspect terms extraction and an opinion attention for opinion terms extraction. Each of them aims to learn a general prototype vector, a token-level feature vector and a token-level attention score for each word in the sentence. The feature vector and attention score measure the extent of correlation between each input token and the prototype through a tensor operator, where a token with high score indicates a high chance of being an aspect or opinion.
- To capture direct relations between aspect and opinion terms, e.g., the $A \xrightarrow{xcomp} B$ relation shown in Figure 5.1, the aspect and opinion attention are coupled in learning such that the learning of each attention is affected by the other. This helps to double-propagate information between them.
- To further capture indirect relations among aspect and opinion terms, e.g., the $A \xrightarrow{nsubj} C \xleftarrow{acl} B$ relation shown in Figure 5.1, we construct a memory network with multiple layers to update the learned information for better extraction.

Figure 5.3 provides an illustration on how the memory network is able to learn the token-level interactions. Suppose the aspect attention already attended to *burger* and *tastes* in the first layer. The features of these two words will be incorporated to update the prototype vector in the second layer. In this way, the aspect attention is able to select *Fish* in the second iteration, which simulates the information propagation in the syntactic structure. The overall architecture of MNCA is shown in Figure 5.3, where each block shows the computation of a single layer with the input being $\mathbf{X}$ and 3-dimensional tensors $\{\mathbf{G}^a, \mathbf{D}^a, \mathbf{G}^p, \mathbf{D}^p\}$. Next, we illustrate each component in details.

5.3.1 Attention with Tensor Operator

A basic unit of MNCA is a pair of attentions: aspect attention and opinion attention. Different from traditional attentions which are used for generating a weighted sum of

Figure 5.2: An example of second layer of MNCA.

Figure 5.3: Architecture of dual propagation memory networks for aspect & opinion terms extraction.

the input to represent the sentence-level information, we use attention to identify the possibility of each token being an aspect or opinion term. As shown in Figure 5.4.a, given a sentence with pre-trained word embeddings $\mathbf{X}=[\mathbf{x}_1, \dots, \mathbf{x}_{n_i}]$, we first apply Gated Recurrent Unit (GRU) [18] to obtain a memory matrix $\mathbf{H}=[\mathbf{h}_1, \dots, \mathbf{h}_{n_i}]$, where $\mathbf{h}_j \in \mathbb{R}^d$ is a feature vector for j -th token considering its context.

In the aspect attention, we first generate a prototype vector $\mathbf{u}^a$ which can be viewed as a general feature representation for aspect terms. This aspect prototype will guide the model to attend to the most relevant tokens (most likely aspect words).¹ Given $\mathbf{u}^a$ and $\mathbf{H}$, the model scans the input sequence and computes an attention vector $\mathbf{r}_j^a$ and an attention score α_j^a for the j -th token. To obtain $\mathbf{r}_j^a$, we first compute a composition vector $\beta_j^a \in \mathbb{R}^K$ that encodes the extent of correlations between $\mathbf{h}_j$ and prototype vector $\mathbf{u}^a$ through a tensor operator:

$$\beta_j^a = \tanh(\mathbf{h}_j^\top \mathbf{G}^a \mathbf{u}^a), \quad (\text{Eq. 5.1})$$

where $\mathbf{G}^a \in \mathbb{R}^{K \times d \times d}$ is a 3-dimensional tensor. Motivated by [104], a tensor operator could be viewed as multiple bilinear matrices that could model more complicated compositions between 2 units. Here, $\mathbf{G}^a$ could be decomposed into K slices, where each slice $\mathbf{G}_k^a \in \mathbb{R}^{d \times d}$ is a bilinear term that interacts with 2 vectors and captures one type of composition, e.g., a specific syntactic relation. Hence $\mathbf{h}_j^\top \mathbf{G}^a \mathbf{u}^a \in \mathbb{R}^K$ inherits K different kinds of compositions between $\mathbf{h}_j$ and $\mathbf{u}^a$ that indicates complicated correlations between each input token and the aspect prototype. Then $\mathbf{r}_j^a$ is obtained from β_j^a via a GRU network:

$$\mathbf{r}_j^a = (1 - z_j^a) \odot \mathbf{r}_{j-1}^a + z_j^a \odot \tilde{\mathbf{r}}_j^a, \quad (\text{Eq. 5.2})$$

where $g_j^a = \sigma(\mathbf{W}_g^a \mathbf{r}_{j-1}^a + \mathbf{U}_g^a \beta_j^a)$, $z_j^a = \sigma(\mathbf{W}_z^a \mathbf{r}_{j-1}^a + \mathbf{U}_z^a \beta_j^a)$, and $\tilde{\mathbf{r}}_j^a = \tanh(\mathbf{W}_r^a (g_j^a \odot \mathbf{r}_{j-1}^a) + \mathbf{U}_r^a \beta_j^a)$. This step helps to encode sequential context information into the attention vector

¹We randomly initialize $\mathbf{u}^a$ from a uniform distribution: $\mathbf{u}^a \sim U[-0.2, 0.2] \in \mathbb{R}^d$, which is then trained and updated iteratively.

Figure 5.4: Independent attentions vs coupled attentions with tensor operator.

$\mathbf{r}_j^a \in \mathbb{R}^K$. Indeed, many aspect terms consist of multiple tokens, and exploiting context information helps their predictions. For simplicity, we use $\mathbf{r}_j^a = f_{\text{GRU}}(\mathbf{r}_{j-1}^a, \beta_j^a; \Theta^a)$ where $\Theta^a = \{\mathbf{W}_g^a, \mathbf{U}_g^a, \mathbf{W}_z^a, \mathbf{U}_z^a, \mathbf{W}_r^a, \mathbf{U}_r^a\}$ to denote (Eq. 5.2).

An attention score α_j^a for token w_j is then computed as

$$\alpha_j^a = \exp(\mathbf{e}_j^a) / \sum_k \exp(\mathbf{e}_k^a), \quad (\text{Eq. 5.3})$$

where α_j^a denotes the j -th element of the vector α^a , similar for $\mathbf{e}_j$. Here $\mathbf{e}_j^a = \langle \mathbf{v}^a, \mathbf{r}_j^a \rangle$. Since $\mathbf{r}_j^a$ is a correlation feature vector, $\mathbf{v}^a \in \mathbb{R}^K$ can be deemed as a weight vector that weighs each feature accordingly. Hence, α_j^a becomes a normalized score, where a higher score indicates higher correlation with the prototype, and higher chance of being attended. The procedure for opinion attention is similar. In the subsequent sections, we use a superscript p to denote the opinion attention.

5.3.2 Coupled Attentions for Dual Propagation

As discussed in previous sections, a crucial issue for co-extraction of aspect and opinion terms is how to fully exploit the relations between aspect terms and opinion terms such that the information can be propagated to each other to assist final predictions. However, independent learning of the attentions in Section 5.3.1 fails to utilize their relations. Here, we propose to couple the learning of the two attentions such that information of each attention can be dually propagated to the other. As shown in 5.4.b, Different from a single attention, the prototype to be fed into each attention module becomes a pair of vectors $\{\mathbf{u}^a, \mathbf{u}^p\}$, and the tensor operator in (Eq. 5.1) becomes a set of tensors $\{\mathbf{G}^a, \mathbf{D}^a, \mathbf{G}^p, \mathbf{D}^p\}$:

$$\beta_j^a = \tanh([\mathbf{h}_j^\top \mathbf{G}^a \mathbf{u}^a : \mathbf{h}_j^\top \mathbf{D}^a \mathbf{u}^p]), \text{ and } \beta_j^p = \tanh([\mathbf{h}_j^\top \mathbf{G}^p \mathbf{u}^a : \mathbf{h}_j^\top \mathbf{D}^p \mathbf{u}^p]), \quad (\text{Eq. 5.4})$$

where $[:]$ denotes concatenation of vectors. Intuitively, $\mathbf{G}^a$ or $\mathbf{D}^p$ is to capture the K syntactic relations within aspect terms or opinion terms themselves, while $\mathbf{G}^p$ and $\mathbf{D}^a$ are to capture syntactic relations between aspect terms and opinion terms, which captures the dual-propagation between aspects and opinions. Note that β_j^a and β_j^p , both of which are of $2K$ dimensions will go through the same procedure as (Eq. 5.2) and (Eq. 5.3) to produce $\mathbf{r}_j^a, \mathbf{r}_j^p \in \mathbb{R}^{2K}$ as the hidden representations for $\mathbf{h}_j$ w.r.t. aspect and opinion attention, respectively.

5.3.3 Memory Network

The coupled attentions aim to capture the direct relations between aspect terms and opinion terms, but fail to exploit the indirect relations among them, such as the $A \xrightarrow{nsbj} C \xleftarrow{acl} B$ relation shown in Figure 5.1. To address this issue, we integrate the coupled attentions into a memory network such that the information learned from the attentions could be updated and used for better extraction. The memory network consists of multiple layers of coupled attentions. For each layer $t+1$ as shown in Figure 5.3, the prototype

vectors $\mathbf{u}_{t+1}^a$ and $\mathbf{u}_{t+1}^p$ are updated based on the prototype vectors in the previous layer $\mathbf{u}_t^a$ and $\mathbf{u}_t^p$ to incorporate more feasible representations for aspect or opinion terms through

$$\mathbf{u}_{t+1}^a = \tanh(\mathbf{Q}^a \mathbf{u}_t^a) + \mathbf{o}_t^a, \text{ and } \mathbf{u}_{t+1}^p = \tanh(\mathbf{Q}^p \mathbf{u}_t^p) + \mathbf{o}_t^p, \quad (\text{Eq. 5.5})$$

where $\mathbf{Q}^a, \mathbf{Q}^p \in \mathbb{R}^{d \times d}$ are recurrent transformation matrices to be learned, and $\mathbf{o}_t^a, \mathbf{o}_t^p$ are accumulated vectors computed via

$$\mathbf{o}_t^a = \sum_j \alpha_t^a \mathbf{h}_j, \text{ and } \mathbf{o}_t^p = \sum_j \alpha_t^p \mathbf{h}_j. \quad (\text{Eq. 5.6})$$

Intuitively, $\mathbf{o}_t^a$ and $\mathbf{o}_t^p$ are dominated by the input feature vectors $\{\mathbf{h}_j\}$'s with higher attention scores. Therefore, $\mathbf{o}_t^a$ and $\mathbf{o}_t^p$ will approach to the attended feature vectors of aspect or opinion words. In this way, $\mathbf{u}_{t+1}^a$ (or $\mathbf{u}_{t+1}^p$) incorporates most probable aspect (or opinion) terms, which in turn will be used to interact with $\{\mathbf{h}_i\}$'s at layer $t+1$ to learn more precise token representations, attention scores, and sentence representations for selecting other non-obvious target tokens. At the last layer T , after generating all the $\{\mathbf{r}_{T,j}^a\}$'s and $\{\mathbf{r}_{T,j}^p\}$'s, we compute two 3-dimensional label vectors $\mathbf{y}_j^a$ and $\mathbf{y}_j^p$ as follows,

$$\mathbf{y}_j^a = \text{softmax}(\mathbf{W}^a \mathbf{r}_{T,j}^a), \text{ and } \mathbf{y}_j^p = \text{softmax}(\mathbf{W}^p \mathbf{r}_{T,j}^p), \quad (\text{Eq. 5.7})$$

where $\mathbf{W}^a, \mathbf{W}^p \in \mathbb{R}^{3 \times 2K}$ are transformation matrices for aspect and opinion prediction, respectively, and $\mathbf{y}_j^a$ denotes the probabilities of $\mathbf{h}_j$ being BA, IA and N, while $\mathbf{y}_j^p$ denotes the probabilities of $\mathbf{h}_j$ being BP, IP and N. For training, we define the loss function as follows,

$$\mathcal{L} = \sum_{j=1}^{n_i} \sum_{m \in \{a,p\}} \ell(\hat{\mathbf{y}}_j^m, \mathbf{y}_j^m), \quad (\text{Eq. 5.8})$$

where $\ell(\cdot, \cdot)$ is the cross-entropy loss, and $\hat{\mathbf{y}}_j^m \in \mathbb{R}^3$ is a one-hot vector representing ground-truth label for j -th token w.r.t. aspect or opinion. For testing or making predictions, the final label for each token j is produced by comparing the values in $\mathbf{y}_j^a$ and $\mathbf{y}_j^p$. If both of them are N, then the label is N. If only one of them is N, we pick the other one as the label. Otherwise, the label is the one with the largest value.

5.4 Multi-task Memory Network for Extraction & Categorization

We further extend MNCA to deal with category-specific aspect and opinion terms extraction by integrating multi-task learning strategy. The proposed multi-task memory network consists of four main components: 1) **Category-specific MNCA** to co-extract aspect and opinion terms for each category, 2) **Shared Tensor Decomposition** to model the commonalities of syntactic relations among different categories by sharing the tensor parameters, 3) **Context-aware Multi-task Feature Learning**, which aims to jointly learn features among categories through constructing context-aware task similarity matrices, and 4) **Auxiliary Task**, which creates an auxiliary task to predict overall sentence-level category labels to assist token-level prediction tasks. In the following section, we present the 4 components of MTMN in detail.

5.4.1 Category-specific MNCA

We use MNCA as the base classifier in MTMN for aspect and opinion terms co-extraction for each category c . According to Section 5.3, we apply the procedure of MNCA for each category c by denoting each variable with the subscript c :

$$\beta_{c[j]}^a = \tanh([\mathbf{h}_j^\top \mathbf{G}_c^a \mathbf{u}_c^a : \mathbf{h}_j^\top \mathbf{D}_c^a \mathbf{u}_c^p]), \text{ and } \beta_{c[j]}^p = \tanh([\mathbf{h}_j^\top \mathbf{G}_c^p \mathbf{u}_c^a : \mathbf{h}_j^\top \mathbf{D}_c^p \mathbf{u}_c^p]), \text{ (Eq. 5.9)}$$

where $\mathbf{G}_c^a, \mathbf{G}_c^p, \mathbf{D}_c^a, \mathbf{D}_c^p \in \mathbb{R}^{K \times d \times d}$. Then we obtain $\mathbf{r}_{c[j]}^a$ and $\mathbf{r}_{c[j]}^p$ as the hidden representations for $\mathbf{h}_j$ w.r.t. aspect and opinion of category c , respectively. Normalized attention scores for $\mathbf{h}_j$ for each category c are computed as

$$\alpha_{c[j]}^a = \exp(\mathbf{e}_{c[j]}^a) / \sum_k \exp(\mathbf{e}_{c[k]}^a), \text{ and } \alpha_{c[j]}^p = \exp(\mathbf{e}_{c[j]}^p) / \sum_k \exp(\mathbf{e}_{c[k]}^p), \text{ (Eq. 5.10)}$$

The overall representations of the sentence for category c in terms of aspects and opinions, denoted by $\mathbf{o}_c^a$ and $\mathbf{o}_c^p$, respectively, are then computed using (Eq. 5.6), which are further

used to produce the prototype vectors $\mathbf{u}_{c,t+1}^a, \mathbf{u}_{c,t+1}^p$ in the next layer using (Eq. 5.5). At the last layer T , after generating all the $\{\mathbf{r}_{c[j]}^a\}$'s and $\{\mathbf{r}_{c[j]}^p\}$'s, for each category c^2 , we compute two 3-dimensional label vectors $\mathbf{y}_{c[j]}^a$ and $\mathbf{y}_{c[j]}^p$ as follows,

$$\mathbf{y}_{c[j]}^a = \text{softmax}(\mathbf{W}^a \mathbf{r}_{c[j]}^a), \text{ and } \mathbf{y}_{c[j]}^p = \text{softmax}(\mathbf{W}^p \mathbf{r}_{c[j]}^p). \quad (\text{Eq. 5.11})$$

For training, we define the loss function as follows,

$$\mathcal{L}_{\text{tok}} = \sum_c \sum_{j=1}^{n_i} \sum_{m \in \{a,p\}} \ell(\hat{\mathbf{y}}_{c[j]}^m, \mathbf{y}_{c[j]}^m). \quad (\text{Eq. 5.12})$$

For testing, we generate a label for each token j as follows. We first produce a label $\mathbf{y}_{c[j]}$ for category c on the j -th token by comparing the largest value in $\mathbf{y}_{c[j]}^a$ and $\mathbf{y}_{c[j]}^p$ using the same method as MNCA. We then generate the final label on the j -th token by integrating $\mathbf{y}_{c[j]}$'s across all the categories, which is similar to multi-label classification, because some word might belong to multiple categories.

If we directly apply the above formulation to extract aspect terms and opinion terms for each category independently, the result is not satisfactory as will be shown in the experiments. This is because in our proposed finer-grained extraction problem, training data for each specific category becomes too sparse to learn precise predictive models if extractions for different categories are considered independently. In the following sections, we introduce how to encode multi-task learning techniques to make the dual propagation memory network effective for finer-grained sentiment analysis.

5.4.2 Shared Tensor Decomposition

As described in the previous section, for each category c , there are four tensor operators $\mathbf{G}_c^a, \mathbf{G}_c^p, \mathbf{D}_c^a$, and $\mathbf{D}_c^p$, each of which is in $\mathbb{R}^{K \times d \times d}$, to model the complex token interactions. However, when the number of categories increases, the parameter size may be

²We omit the subscript T for the ease of notation

Figure 5.5: The architecture of each non-output layer used in MTMNs.

very large. As a result, available training data may be too sparse to learn the precise parameters. Therefore, instead of learning the tensors for each category independently, we assume that interactive relations among tokens are similar across categories. Therefore, we propose to learn a low-rank shared information among the tensors through collective tensor factorization as shown in the architecture of each non-output layer in Figure 5.5. Specifically, let $\mathbf{G}^a \in \mathbb{R}^{C \times K \times d \times d}$ be the concatenation of all the $\{\mathbf{G}_c^a\}$'s, and denote by $\mathbf{G}_k^a = \mathbf{G}^a_{[:,k, :, :] } \in \mathbb{R}^{C \times d \times d}$ the collection of k -th bi-linear interaction matrices across C tasks for aspect attention. The same also applies to $\mathbf{G}^p$ and $\mathbf{G}_k^p$ for opinion attention. Factorization is performed on each $\mathbf{G}_k^a$ and $\mathbf{G}_k^p$, respectively, via

$$\mathbf{G}_{k[c, :, :]}^a = \mathbf{Z}_{k[c, :]}^a \mathbf{G}_k^a, \text{ and } \mathbf{G}_{k[c, :, :]}^p = \mathbf{Z}_{k[c, :]}^p \mathbf{G}_k^p, \quad (\text{Eq. 5.13})$$

where $\mathbf{G}_k^a, \mathbf{G}_k^p \in \mathbb{R}^{m \times d \times d}$ are shared factors among all the tasks with $m < C$, while $\mathbf{Z}_k^a, \mathbf{Z}_k^p \in \mathbb{R}^{C \times m}$ with each row $\mathbf{Z}_{k[c, :]}^a$ and $\mathbf{Z}_{k[c, :]}^p$ being specific factors for category c . The shared factors can be considered as m latent basis interactions, where the original k -th bi-linear relation matrix $\mathbf{G}_{k[c, :, :]}^a$ ($\mathbf{G}_{k[c, :, :]}^p$) for c is the linear combination of the latent basis interactions. The same approach also applies to the tensors $\{\mathbf{D}_c^a\}$'s and $\{\mathbf{D}_c^p\}$'s. In this

way, we reduce the parameter dimensions by enforcing sharing within a small number of latent interactions.

5.4.3 Context-aware Multi-task Feature Learning

Besides jointly decomposing tensors of syntactic relations across categories, in this section, we further exploit similarities between categories or tasks³ to learn more powerful features for each token and each sentence. Consider the following motivating example, “FOOD#PRICE” is more similar to “DRINK#PRICE” than “SERVICE#GENERAL” because the first two categories may share some common aspect/opinion terms, such as *expensive*. Therefore, by representing each task in a form of distributed vector, we can directly compute their similarities to facilitate knowledge sharing. Based on this motivation, we aim to update features $\tilde{\mathbf{r}}_c^a$ (or $\tilde{\mathbf{r}}_c^p$) from $\mathbf{r}_c^a$ (or $\mathbf{r}_c^p$) by integrating task relatedness. Specifically, at a layer t , suppose $\mathbf{u}_{c,t}^a$, and $\mathbf{u}_{c,t}^p$ are the updated prototype vectors passed from the previous layer, they can be used to represent task c . Because $\mathbf{u}_{c,t}^a$ and $\mathbf{u}_{c,t}^p$ are learned interactively with the category-specific sentence representations $\mathbf{o}_c^a$ ’s and $\mathbf{o}_c^p$ ’s of the previous $t-1$ layers, respectively. Let $\mathbf{U}^a$ and $\mathbf{U}^p \in \mathbb{R}^{d \times C}$ denote the matrices consisting of $\mathbf{u}_c^a$ and $\mathbf{u}_c^p$ as a column vector, respectively, then the task similarity matrices, $\mathbf{S}^a$ and $\mathbf{S}^p$, in terms of aspects and opinions can be computed as follows,

$$\mathbf{S}^a = q(\mathbf{U}^{a\top} \mathbf{U}^a), \text{ and } \mathbf{S}^p = q(\mathbf{U}^{p\top} \mathbf{U}^p), \quad (\text{Eq. 5.14})$$

where $q(\cdot)$ is the softmax function carried in a column-wise manner so that the similarity scores between a task and all the tasks sum up to 1. The similarity matrices $\mathbf{S}^a$ and $\mathbf{S}^p$ are then used to refine feature representation of each token for each task by incorporating feature representations from related tasks:

$$\tilde{\mathbf{r}}_{c,[j]}^a = \sum_{c'=1}^C \mathbf{S}_{cc'}^a \mathbf{r}_{c',[j]}^a, \text{ and } \tilde{\mathbf{r}}_{c,[j]}^p = \sum_{c'=1}^C \mathbf{S}_{cc'}^p \mathbf{r}_{c',[j]}^p, \quad (\text{Eq. 5.15})$$

³We interchangeably use the terms “task” and “category” in the rest of this chapter.

where $\mathbf{r}_{c',[j]}^a$ and $\mathbf{r}_{c',[j]}^p$ denote the j -th column of the matrix $\mathbf{r}_{c'}^a$ and $\mathbf{r}_{c'}^p$, respectively. Similarly, we refine feature representation of each sentence for each task as follows,

$$\tilde{\mathbf{o}}_c^a = \sum_{c'=1}^C \mathbf{S}_{cc'}^a \mathbf{o}_{c'}^a, \text{ and } \tilde{\mathbf{o}}_c^p = \sum_{c'=1}^C \mathbf{S}_{cc'}^p \mathbf{o}_{c'}^p. \quad (\text{Eq. 5.16})$$

Regarding update of the prototype vectors, we replace $\mathbf{o}_c^a$ and $\mathbf{o}_c^p$ by $\tilde{\mathbf{o}}_c^a$ and $\tilde{\mathbf{o}}_c^p$, respectively. This context-aware multi-task architecture is also shown in Figure 5.5. Note that this feature sharing among different tasks is context-aware because $\mathbf{U}^a$ and $\mathbf{U}^p$ are category representations depending on each sentence. This means that different sentences might indicate different task similarities. For example, when *cheap* is presented, it might increase the similarity between “FOOD#PRICES” and “RESTAURANT#PRICES”. As a result, $\tilde{\mathbf{r}}_{c[j]}^a$ for task c could incorporate more information from task c' if c' has higher similarity score indicated by $\mathbf{S}_{cc'}^a$.

5.4.4 Auxiliary Task

As shown in previous sections, MTMNs could produce sentence-level feature representations. Thus, to better address the data sparsity issue, we aim to use additional global information on categories in the sentence level. Consider the following motivating example, if we know the sentence “*The soup is served with nice portion, the service is prompt*” belongs to category “DRINKS#STYLE.OPTIONS” and “SERVICE#GENERAL”, we can infer that some words in the sentence should belong to one of these two categories. To make use of this information, we construct an auxiliary task to predict the categories of a sentence. From training data, sentence-level labels can be automatically obtained by integrating tokens’ labels. Therefore, besides the token loss (Eq. 5.12) for our target token-level prediction tasks, we also define sentence loss for the auxiliary task. Note that the learning of the target task (terms extraction) and auxiliary task (multi-label classification on sentences) are not independent. On the one hand, the global sentence

Figure 5.6: The architecture of the output layer used in MTMNs.

information helps the attentions to select category-relevant tokens. On the other hand, if the attentions are able to attend to target terms, the output context representation will filter out irrelevant noise, which helps the overall sentence prediction. To be specific, as shown in Figure 5.6, for category c , we define $\tilde{\mathbf{o}}_c = [\tilde{\mathbf{o}}_c^a; \tilde{\mathbf{o}}_c^p] \in \mathbb{R}^{2d}$ the final representation for the sentence, and generate the output using the softmax function,

$$\mathbf{l}_c = \text{softmax}(\mathbf{W}_c \tilde{\mathbf{o}}_c), \quad (\text{Eq. 5.17})$$

where $\mathbf{W}_c \in \mathbb{R}^{2 \times 2d}$, and $\mathbf{l}_c \in \mathbb{R}^2$ indicates the probability of the sentence belonging to category c or not. The loss of the auxiliary task is defined as $\mathcal{L}_{\text{sen}} = \sum_c \ell(\hat{\mathbf{l}}_c, \mathbf{l}_c)$, where $\ell(\cdot)$ is the cross-entropy loss, and $\hat{\mathbf{l}}_c \in \{0, 1\}^2$ is the ground truth using one-hot encoding indicating whether category c is presented for the sentence. By incorporating the loss of the auxiliary task, the final objective for MTMNs is written as $\mathcal{L} = \mathcal{L}_{\text{sen}} + \mathcal{L}_{\text{tok}}$, where $\mathcal{L}_{\text{tok}}$ is defined in (Eq. 5.12).

Dataset	Description	Training		Test		Total	
		text	tuple	text	tuple	text	tuple
S1	SemEval-15 Restaurant	1,315	1,654	685	845	2,000	2,499
S2	SemEval-16 Restaurant	2,000	2,507	676	859	2,676	3,366
S3	SemEval-14 Laptop	3,045	1,974	800	545	3,845	2,519
S4	SemEval-14 Restaurant	3,041	-	800	-	3,841	-

Table 5.1: Dataset description.

5.5 Experiment

5.5.1 Dataset & Experimental Setup

The experiments are conducted on four benchmark datasets from subtask 1 in SemEval Challenge 2015 task 12 [89], SemEval Challenge 2016 task 5 [88], laptop and restaurant dataset in SemEval Challenge 2014 task 4 [90], which are denoted by S1, S2, S3 and S4 respectively. Note that S1 and S2 are both reviews in restaurant domain. We use the same term-level aspect-opinion annotations as in Section 4.4 for S3 and S4. We also manually annotate opinion terms for S1 and S2. To facilitate our experiment, we additionally annotate category labels on target terms for S3, while the aspect term categories for S1 and S2 are provided by SemEval. The statistics of each dataset is shown in Table 5.1 where *text* and *tuple* represent the number of sentences and the number of tuples consisting of an aspect term and its corresponding category label, respectively. Each sentence may contain multiple aspect terms with more than one categories. The aspect categories are shown in Table 5.2.⁴ For S1 and S2, an aspect category is defined as the combination of an entity and an attribute, e.g., “FOOD#PRICES”. There are in total 12 categories. For S3, an aspect category is an entity.

Follow Section 4.4, we first obtain word embeddings by applying *word2vec* on Yelp Challenge dataset and electronic domain in Amazon reviews for restaurant and laptop datasets, respectively. We set the dimension of word embeddings to be 150 and the

⁴We filter out some categories with very few target terms and remove the corresponding sentences.

Restaurant		Laptop
Entity Labels	Attribute Labels	Entity Labels
1. RESTAURANT 2. FOOD 3. DRINKS 4. AMBIENCE 5. SERVICE 6. LOCATION	A. GENERAL B. PRICES C. QUALITY D. STYLE_OPTIONS E. MISCELLANEOUS	1. LAPTOP 2. DISPLAY 3. KEYBOARD 4. MOUSE 5. BATTERY 6. GRAPHICS 7. HARD_DISK 8. MULTIMEDIA_DEVICES 9. SOFTWARE 10. OS 11. SUPPORT 12. COMPANY

Table 5.2: Aspect Categories for two domains.

dimension after GRU transformation to be 50. We use two layers of memory network for experiments. For each layer, the number of bi-linear interactions for the 3-dimensional tensors is 20 ($K=20$), and tensor factorization for MTMN operates with $m=5$ for S1 and S2, and $m=8$ for S3. We apply partial dropout at 0.5 to chosen parameters (non-recurrent parameters of GRU) to avoid overfitting. For MNCA, we fix the learning rate to be 0.07 for S1, S2, S4, and 0.1 for S3. For MTMN, the training is carried with *rmsprop* with the initial value at 0.001 and decayed with rate 0.9. The trade-off parameter λ is set to be 1.0. All the hyper-parameters are chosen according to cross-validation.

5.5.2 Result & Analysis for Aspect and Opinion Terms Extraction

In this section, we illustrate our experimental results of the proposed MNCA model for simply aspect terms and opinion terms extraction and show the state-of-the-art performances compared with various baseline models listed in the following:

EliXa, NLANG, DLIREC, IHS_RD: the top performing systems for S1, S2, S3, S4 in corresponding SemEval Challenges.

LSTM: an LSTM network built on top of word embeddings proposed by [66]. The settings are the same as Section 4.4.

WDEmb: the model proposed by [125] using word and dependency path embeddings combined with linear context embedding features, dependency context embedding features as CRF input.

	S1		S2		S3		S4	
Model	AS	OP	AS	OP	AS	OP	AS	OP
EliXa	70.04	-	-	-	-	-	-	-
NLANG	67.11	-	72.34	-	-	-	-	-
DLIREC	-	-	-	-	73.78	-	84.01	-
IHS_RD	63.12	-	-	-	74.55	-	79.62	-
LSTM	64.30	66.43	68.43	72.04	72.73	74.98	81.15	80.22
WDEmb	69.12	-	-	-	74.68	-	84.31	-
WDEmb*	69.73	-	-	-	75.16	-	84.97	-
RNCRF	67.06	66.90	69.09	75.79	76.83	76.76	84.05	80.93
RNCRF*	67.74	67.62	69.74	76.15	78.42	79.44	84.93	84.11
MNCA	70.73	73.68	75.21	77.90	77.80	80.17	85.29	83.18

Table 5.3: Comparison results in terms of F_1 scores. AS (OS) refers to aspect (opinion) terms extraction.

RNCRF: the joint model with CRF and recursive neural network introduced in Chapter 4, which has been shown to outperform CRFs with hand-crafted features.

WDEmb*, **RNCRF***: the corresponding models with additional human-engineered linguistic features.

The comparison results in terms of F_1 scores are shown in Table 5.3. We report results for both aspect terms extraction (AS) and opinion terms extraction (OP) for all the four datasets. To make fair comparisons, we use the same corpus as in LSTM, RNCRF, RNCRF* for training word embeddings, and same training set with both aspect and opinion labels. Among deep-learning-based models, the models that combine neural network with CRF (i.e., WDEmb and RNCRF) perform better than LSTM because of the incorporation of dependency structure. It is clear that MNCA achieves the state-of-the-art results for most of the time without any pre-extracted linguistic/syntactic information. Specifically, MNCA outperforms WDEmb by 1.61%, 3.12% and 0.98% respectively for S1, S3 and S4, and RNCRF by 3.67%, 6.12%, 0.97% and 1.24%, on S1, S2, S3 and S4, respectively, for aspect extraction. Even compared with the deep models with additional hand-crafted features, i.e., WDEmb* and RNCRF*, MNCA still gets improvements for most of the time. Moreover, the improvements over RNCRF and

		S1		S2		S3		S4	
		AS	OP	AS	OP	AS	OP	AS	OP
Layer	1	69.27	69.56	75.10	78.94	77.28	78.12	84.90	81.85
	2	70.73	73.68	75.21	77.90	77.80	80.17	85.29	83.18
	3	69.78	71.95	73.82	76.80	77.24	79.29	84.41	82.38
Setting	ASL	69.53	-	73.87	-	76.45	-	84.38	-
	ASL+OPL	69.49	72.73	74.09	77.03	77.05	79.66	84.14	82.10
	MNCA	70.73	73.68	75.21	77.90	77.80	80.17	85.29	83.18

Table 5.4: Comparisons under varying layers and settings.

RNCRF* are all significant ($p < 0.01$), except for the aspects extraction on S1 and S2 over RNCRF*. Note that besides linguistic features, WDEmb* and RNCRF* also require dependency parsers to perform the task. Therefore, MNCA is more effective and simpler to implement.

To show the effect of the number of layers, we present experimental results varying the number of layers in Table 5.4. The best results are obtained with 2 layers. With only one layer, the results for aspect extraction are 1.46%, 0.11%, 0.52% and 0.39% inferior than the best scores on S1, S2, S3 and S4, respectively, but they are still comparable with other baselines shown in Table 5.3. Similar observations can be found for the results with 3 layers. This shows that MNCA with 2 layers is enough to exploit most of the relations among input tokens.

We also conducted experiments to explicitly show the advantage of coupling the learning of aspect and opinion attentions. The second part in Table 5.4 specifies different settings of the model. ASL refers to the multi-layer network with only aspect attention and is trained with aspect labels only. We can see that even without opinion labels, the network still proves comparable and even superior than deep models without linguistic features for aspect terms extraction shown in Table 5.3. This shows that multi-layer attentions with tensors is advantageous for exploiting interactions. ASL+OPL in Table 5.4 trains the aspect attention and opinion attention independently using (Eq. 5.1) where each attention predicts one of the three labels. The results of ASL+OPL in terms

of aspect extraction are similar to ASL, which shows that the additional opinion labels have little effect on aspect extraction if they are not interactively trained. By coupling the aspect and opinion attentions, MNCA achieves the best performance.

As a core component, an attention computes a score for each token to indicate its correlation with the corresponding prototype. We visualize the actual attention scores for the tokens of 4 sentences in Figure 5.7. The y-axis represents the scores before normalization which can be positive or negative, but only the magnitude matters. Higher scores mean larger correlations with the aspect/opinion prototype. As the aspect and opinion attention have different sets of parameters, the scores can correspond to different ranges of the values. Tokens in purple (blue) are the ground-truth aspect (opinion) terms. Obviously, purple tokens correspond to large scores for aspect extraction (purple bars with large values), and blue tokens correspond to large scores for opinion extraction (blue bars with large values). All the other non-relevant terms have lower scores. This shows that our model is able to extract terms of interest.

As mentioned previously, MNCA is able to extract target terms without any dependency parser, and hence does not depend on the quality of the parsing results. To show that, we pick a few example reviews from the test datasets as presented in Table 5.5. The left and right column show the prediction results from the proposed model and RNCRF, respectively, where predicted opinions are made *italic*, and aspects are “quoted”. Obviously, the listed reviews are not formal enough to be parsed correctly. Hence, RNCRF fails to extract some of the targets, unlike MNCA which identifies all possible target terms.

To show the robustness of MNCA, we provide two sensitivity studies on word embedding dimensions and the number of different interactions within a 3-dimensional tensor on S4 in Figure 5.8.a and Figure 5.8.b. From the plot, we can see that the performances for both aspect and opinion terms extraction are relatively stable when varying word embedding dimensions, with the highest scores achieved at 200. For the number of tensor

Figure 5.7: Visualization of attention weights for different tokens within a sequence.

Prediction with MNCA	Prediction with RNCRF
also <i>stunning</i> “colors” and <i>speedy</i>	also <i>stunning colors</i> and <i>speedy</i>
Only 2 “usb ports” ... seems kind of <i>limited</i>	Only 2 “usb ports” ... seems kind of limited
<i>strong</i> “build” though which really adds to its “durability”	<i>strong</i> “build” though which really adds to its durability
Save room for “deserts” - they’re to <i>die for</i>	Save room for “deserts” - they’re to die for
You must try “Odessa stew” or “Rabbit stew”; “salads” - all <i>good</i>	You must try “Odessa stew or Rabbit stew”; salads - all good

Table 5.5: Prediction comparison between MNCA and RNCRF

interactions, the model attains the best performance at 20 for aspect extraction and 10 for opinion extraction.

5.5.3 Result & Analysis for Extraction and Categorization

This section introduces the effectiveness of the proposed MTMN for category-specific aspect and opinion terms extraction. The results are shown for the first three datasets because we do not manually label the category information for S4. Moreover, some

5.8.a: On word embedding.

5.8.b: On tensor interaction.

5.8.c: On tensor factorization.

Figure 5.8: Sensitivity studies for data S1 and S4.

sentences in S3 whose categories are rarely seen are removed to facilitate our experiment.

We conduct comparisons with the following baseline models:

NLANG: The best system for both SemEval-15 and SemEval-16 for the proposed task.

IHS-RD, XRCE: The second best systems for SemEval-15 and SemEval-16, respectively.

RNCRF+: We modify RNCRF by defining finer-grained categories as labels. This means directly apply RNCRF to a multi-class classification problem with $C \times 5$ classes, as there are 5 classes for each category and there are C categories.

MNCA+: Similar to RNCRF+, we modify MNCA by defining finer-grained categories as labels. Note that the proposed MTMNs can be reduced to this baseline model by only using a single dual propagation memory with $C \times 5$ classes.

MNCA++: MNCA is used to extract all the aspect and opinion terms first, and then a category classification layer is added to classify the extracted terms.

We report the results from top performing systems in the Challenges for S1 and S2. There are no reported results for S3 as the original task is different from ours. Note that original task for S1 and S2 includes two slots: slot 1 for sentence-level aspect category prediction and slot 2 for aspect terms extraction. Moreover, SemEval also evaluated on the pairing of slot 1 and slot 2 by joining them as an additional task that corresponds to the problem we study. However, most of the reported models did not provide feasible methods for the joint prediction of aspect terms and corresponding categories. Instead, they trained the model for slot 2 first and then combined with slot 1. This may fail to capture the relations between target terms and their categories. In order to show the advantage of our model, we modify the existing state-of-the-art deep models for aspect/opinion term extraction to fit our problem settings. Since RNCRF and MNCA both exploit the correlations between aspect terms and opinion terms, which have been shown to be effective for extraction task, a simple idea is to increase the number of classes

	S1				S2				S3			
Model	ASC	OPC	AS	OP	ASC	OPC	AS	OP	ASC	OPC	AS	OP
NLANG	42.90	-	67.11	-	52.61	-	72.34	-	-	-	-	-
IHS_RD	42.72	-	63.12	-	-	-	-	-	-	-	-	-
XRCE	-	-	-	-	48.89	-	61.98	-	-	-	-	-
RNCRF+	54.00	47.86	67.74	67.62	56.04	51.09	69.74	76.15	54.05	58.90	71.87	76.62
MNCA+	57.35	55.70	70.73	73.68	57.83	56.04	75.21	77.90	55.71	62.40	72.42	76.98
MNCA++	53.46	53.94	70.73	73.68	54.05	54.34	75.21	77.90	54.31	62.95	72.42	76.98
MTMN	63.16	59.17	71.31	72.23	65.34	61.44	73.26	76.10	57.06	63.53	69.14	75.76

Table 5.6: Comparison results in terms of F_1 scores. ASC (OPC) refers to category-specific aspect (opinion) terms extraction. AS (OS) refers to aspect (opinion) terms extraction.

to incorporate different categories, e.g., BA becomes $\{BA_c\}$'s for different category c . By increasing the number of classes, the only change to the original model is the dimension of classification matrix. As a result, the modified model should be able to capture the correlations between target terms based on their categories. On the other hand, we also construct another baseline model (denoted by MNCA++) based on MNCA by separating the task into 2 steps. The first step is the same as MNCA for extracting target terms. Then the second step performs category prediction only on the extracted terms.

The comparison results are shown in Table 5.6. It can be seen that MTMN achieves the state-of-the-art performances in category-specific aspect and opinion terms extraction (ASC and OPC). And there is a large gap between the results of MTMN and the other baseline models on S1 and S2. This is because RNCRF+ and MNCA+ can only propagate information between target terms within each category, but fail to explore the relations and commonalities among different categories. The other model MNCA++ performs even poorer, because the training is separated into different stages, similar to the top systems in SemEval Challenges. This separation results in the failure of propagating information from category prediction to target term extraction. The result proves the effectiveness of MTMN for learning shared information among different tasks, as well as the addition of global information to assist extraction. The improvement for S3 is not significant, which might indicate that the category correlations are not obvious in laptop

Different Components	S1				S2			
	ASC (tt)	OPC (tt)	ASC (cv)	OPC (cv)	ASC (tt)	OPC (tt)	ASC (cv)	OPC (cv)
MTMN (C1+C2+C3)	63.16 (1.02)	59.17 (0.27)	63.32	62.98	65.34 (1.20)	61.44 (0.97)	63.65	64.93
C1+C3	61.95 (0.25)	58.57 (0.57)	61.68	59.74	63.30 (0.65)	59.16 (1.24)	61.03	57.17
C2+C3	61.67 (0.67)	55.89 (1.05)	62.25	56.55	60.86 (0.33)	58.68 (0.31)	60.65	59.69
C2+C3*	61.30 (0.59)	55.30 (1.18)	61.64	58.63	62.68 (1.57)	58.93 (0.51)	62.30	60.83
C1+C2	60.67 (0.46)	56.97 (0.78)	62.48	57.62	61.29 (0.50)	58.16 (1.09)	60.43	58.37
C3	60.18 (0.74)	57.03 (0.76)	61.19	58.73	60.57 (0.61)	57.36 (1.19)	61.06	60.04

Table 5.7: Comparison results with reductions.

domain, as can be seen in Table 5.2. Only entity labels make different categories distinct from each other.

Moreover, we also report the results on target terms extraction (AS and OP) by accumulating the aspect/opinion terms that are assigned at least one category by MTMN. It can be seen that MTMN still achieves comparable performances even if the data becomes sparser when adding the category information. On the contrary, the results for RNCRF+, MNCA+ and MNCA++ are obtained using the original models that ignore category labels, which are much easier to obtain high performances.

As have been discussed in the previous sections, the multi-task memory network explores the commonalities and relations among tasks through both tensor sharing and feature sharing, as well as enhances prediction results by incorporating auxiliary labels. To test the effect of each component, we conduct comparison experiments for different combinations of these components as shown in Table 5.7, where C1, C2 and C3 represents separate component for multi-task tensor sharing, context-aware feature sharing and auxiliary task, respectively. Note that for C2+C3*, we use the same tensor across all the tasks. We report both the testing results with standard variation as well as the validation results. Clearly, the inclusion of either feature sharing (C2+C3) or tensor sharing (C1+C3) improves the results for most of the time compared to independent training (C3). Furthermore, tensor sharing is more beneficial than feature sharing. This shows that the commonalities in terms of token interactions are more obvious for different tasks. Moreover, either independent tensors (C2+C3) or the same tensor (C2+C3*)

across tasks does not perform well. This indicates that it is still crucial to explore both the uniqueness and commonality of all the tasks, which are preserved in our proposed model. By comparing the results between C1+C2+C3 and C1+C2, we can see how auxiliary task contributes to the final prediction (2.49% and 4.05% increase for ASC in S1 and S2 respectively). This shows that the global information in the sentence level could enforce the correct predictions of each token within the sentence by reiterating the category information.

To show the robustness of our model, we test it with different dimensions of factorization (m). The results on S1 are shown in Figure 5.8.c. We also provide some examples in Table 5.8 to show what the attentions learn for different categories. The second and third columns show the extracted aspect and opinion terms with corresponding normalized attention scores, respectively. The **bold numbers** denote category indexes. Clearly, MTMN is able to attend to relevant tokens for different categories. Moreover, MTMN could identify the case when specific terms belong to more than one categories.

Sentence	Aspect (score)	Opinion (score)
<i>Excellent</i> food though the interior could use some help.	1 : food (0.77); 2 : interior (0.79)	1 : Excellent (0.56)
The sashimi is always <i>fresh</i> and the rolls are <i>innovative</i> and delicious.	1 : sashimi (0.41), rolls (0.51) 3 : rolls (0.49)	1 : fresh (0.52) 3 : innovative (0.38)
The food was <i>good</i> , the place was <i>clean</i> and <i>affordable</i> .	1 : food (0.70) 2 : place (0.81)	1 : good (0.66); 2 : clean (0.46) 4 : affordable (0.34)
Overall, <i>decent</i> food at a <i>good</i> price, with <i>friendly</i> people .	1 : food (0.70) 6 : people (0.53)	1 : decent (0.31); 5 : good (0.28) 6 : friendly (0.38)
The wine list was <i>expensive</i> , though the staff are not <i>knowledgeable</i> .	6 : staff (0.51) 7 : wine (0.32) list (0.31)	6 : knowledgeable (0.49) 7 : extensive (0.64)
The martinis are <i>amazing</i> and very <i>fairly priced</i> .	8 : martinis (0.42) 9 : martinis (0.53)	8 : fairly (0.21) priced (0.69) 9 : amazing (0.36)
<i>Abulous</i> food , if the front of house staff do n't put you off.	1 : food (0.82); 6 : front (0.26) of (0.10) house (0.23) staff (0.21)	1 : Abulous (0.46)
The food was <i>great</i> and <i>tasty</i> , but the sitting space was too <i>small</i> .	1 : food (0.59) 2 : sitting (0.35) space (0.27)	1 : great (0.36), tasty (0.35) 2 : small (0.32)
I have never been so <i>disgusted</i> by food and service .	1 : food (0.86); 6 : service (0.66)	1, 6 : disgusted (0.50, 0.76)
Despite the <i>confusing</i> mirrors this will be my <i>go-to</i> for Japanese food .	1 : Japanese (0.32) food (0.30) 2 : mirrors (0.35)	1 : go-to (0.50) 2 : confusing (0.39)
Service <i>ok</i> , but <i>unfriendly filthy</i> bathroom .	2 : bathroom (0.79) 6 : service (0.52)	2 : unfriendly (0.33), filthy (0.41) 6 : ok (0.31)

Table 5.8: Examples of attention scores for different categories.

As being discussed, the tensor decomposition of MTMN deals with data sparsity.

Compared with separate tensor for each task, our decomposition exploits sharing among tensor parameters and reduce the complexity. Specifically in the experiments, the tensor for each task has size $20 \times 50 \times 50$. With 12 tasks in total, the resulting parameter size becomes $12 \times 20 \times 50 \times 50$ when there is no sharing. In MTMN, some sharing among the tensors reduces the parameter size to $20 \times (5 \times 50 \times 50 + 20 \times 5)$, which is much smaller than no sharing. Under such condition, our model is able to converge in less than 30 epochs.

5.6 Summary

In this work, we introduce a finer-grained task involving the predictions of both aspect/opinion terms and their corresponding aspect categories, and offer a novel multi-task deep learning model, MTMN, to solve the problem. The model is able to exploit syntactic commonalities and task similarities through attention mechanism. In the end, we demonstrate the effectiveness of our model on three benchmark datasets.

Chapter 6

Domain Adaptation via Syntactic Correspondences

This chapter extends the previously-mentioned single-domain extraction problem to multi-domain settings, where labeled data only exists for the source domain, and the task is to extract aspect terms and opinion terms in the target domain. Most of the existing works for single-domain aspect/opinion terms extraction are not applicable in this case, because the aspect terms from two different domains are usually disjoint, e.g., *laptop* v.s. *restaurant*, leading to large domain shift in the feature vector space. Though unsupervised methods [42, 94] can deal with data with few labels, their performance is unsatisfactory compared with supervised ones.

There have been a number of domain adaptation methods proposed for coarse-grained sentiment classification problems across domains, where an overall sentiment polarity of a sentence or document is being predicted. Nevertheless, there are very few approaches designed for cross-domain fine-grained opinion analysis due to the difficulties in fine-grained adaptation. [63] proposed a bootstrap method based on the TrAdaBoost algorithm [21] to iteratively expand opinion and aspect lexicons in the target domain by exploiting source-domain labeled data and cross-domain common relations between aspect terms and opinion terms. However, their model requires a seed opinion lexicon in the target domain and pre-mined syntactic patterns as a bridge. [25] proposed to use rules to generate

auxiliary supervision on top of a recurrent neural network to learn domain-invariant hidden representation for each word. The performance highly depends on the quality of the manually defined rules and the prior knowledge of a sentiment lexicon. In addition, the recurrent structure fails to capture the syntactic interactions among words intrinsically for opinion extraction.

In this chapter, we propose a novel cross-domain Recursive Neural Network (RvNN) for aspect and opinion terms co-extraction across domains. Our motivations are twofold: 1) The dependency relations capture the interactions among different words. These relations are especially important for identifying aspect terms and opinion terms [94], which are also domain-invariant within the same language. Therefore, they can be used as “pivot” information to bridge the gap between different domains. 2) Inspired by the idea of *structural learning* [1], the success of target task depends on the ability of finding good predictive structures learned from other related tasks, e.g., structural correspondence learning (SCL) [6] for coarse-grained cross-domain sentiment classification. Here, we aim to generate an auxiliary task on dependency relation classification. Different from previous approaches, our auxiliary task and the target extraction task are of heterogeneous label spaces. We aim to integrate this auxiliary task with distributed relation representation learning into a recursive neural network.

Specifically, we generate a dependency tree for each sentence from the dependency parser and construct a unified RvNN that integrates an auxiliary task into the computation of each node. The auxiliary task is to classify the dependency relation for each direct edge in the dependency tree by learning a relation feature vector. To reduce label noise brought by inaccurate parsing trees, we further propose to incorporate an auto-encoder into the auxiliary task to group the relations into different clusters. Finally, to model the sequential context interaction, we develop a joint architecture that combines RvNN with a sequential labeling model for aspect and opinion terms extraction. Extensive experiments are conducted to demonstrate the advantage of our proposed model.

6.1 Problem Definition & Motivation

Problem Definition: Section 3.2 provides a formal definition of the extraction problem in a single-domain with BIO notation. Based on that, we define the problem for unsupervised domain adaptation. Specifically, we are given a set of labeled review sentences denoted with input features from a source domain $\mathcal{D}_S = \{(\mathbf{X}_{S_i}, \mathbf{y}_{S_i})\}_{i=1}^{n_S}$, and a set of unlabeled sentences denoted with input features from a target domain $\mathcal{D}_T = \{\mathbf{X}_{T_j}\}_{j=1}^{n_T}$. Our goal is to predict token-level labels on the target domain $\mathcal{D}_T$.

Motivation: Existing works for cross-domain aspect and/or opinion terms extraction require hand-coded rules and a sentiment lexicon in order to transfer knowledge across domains. For example in Figure 6.1, given a review sentence “*They offer good appetizers*” in the source domain and “*The laptop has a nice screen*” in the target domain. If *nice* has been extracted as a common sentiment word, and “OPINION-amod-ASPECT” has been identified as a common syntactic pattern from the source domain, *screen* could be deduced as an aspect term using the identified syntactic pattern [63]. Similarly, [25] used a set of pre-defined rules based on syntactic relations and a sentiment lexicon to generate auxiliary labels to learn high-level feature representations through a recurrent neural network.

These previous attempts have verified that syntactic information between words, which can be used as a bridge between domains, is crucial for domain adaptation. Moreover, dependency-tree-based RvNN [102] has proven to be effective to learn high-level feature representation of each word by encoding syntactic relations between aspect terms and opinion terms. With the above findings, we propose a novel RvNN named **Recursive Neural Structural Correspondence Network** (RNSCN) to learn high-level representation for each word across different domains. Our model is built upon dependency trees generated from a dependency parser. Different from previous approaches, we do not require

Figure 6.1: An example of two reviews with similar syntactic patterns.

any hand-coded rules or pre-selected pivot features to construct correspondences, but rather focus on the automatically generated dependency relations as the pivots. The model associates each direct edge in the tree with a relation feature vector, which is used to predict the corresponding dependency relation as an auxiliary task.

Note that the relation vector is the key in the model: it associates with the two interacting words and is used to construct structural correspondences between two different domains. Hence, the auxiliary task guides the learning of relation vectors, which in turn affects their correspondingly interactive words. Specifically in Figure 6.1, the relation vector for “amod” is computed from the features of its child and parent word, and is also used to produce the hidden representation of its parent. For this relation path in both sentences, the auxiliary task enforces close proximity for these two relation vectors. This pushes the hidden representations for their parent nodes *appetizers* and *screen* closer to each other, provided that *good* and *nice* have similar representations. In a word, the auxiliary task bridges the gap between two different domains by drawing the words with similar syntactic properties closer to each other.

The relation vectors may be sensitive to the accuracy of the dependency parser. It might harm the learning process when some noises exist for certain relations, especially for non-formal texts. Inspired by the hierarchical structure of dependency relations [12], certain relations with similar functionalities could be grouped together, e.g., *dobj*, *iobj* and *pobj* all indicate objects. We propose to use an auto-encoder to automatically group

Figure 6.2: The architecture of RNSCN-GRU.

these relations in an unsupervised manner. The reconstruction loss helps to reduce label noise by aligning relation features with their intrinsic relation group.

6.2 Methodology

Our proposed model consists of two components. The first component is a Recursive Neural Structural Correspondence Network (RNSCN), and the second component is a sequence labeling classifier. In this paper, we present an implementation for the sequence labeling classifier using Gated Recurrent Unit (GRU). The reason to choose GRU instead of a simple recurrent neural network or LSTM is that GRU is able to deal with long-term dependencies and requires less parameters than LSTM. The resultant deep learning model is denoted by RNSCN-GRU. We also implement Conditional Random Field (CRF) as the sequence labeling classifier, and denote the model by RNSCN-CRF accordingly.

The overall architecture of RNSCN-GRU without auto-encoder is shown in Figure 6.2. The left and right are two example sentences from the source domain and the target domain, respectively. In the first component, RNSCN, an auxiliary task to predict the dependency relation for each direct edge is integrated into a dependency-tree-based RvNN. We generate a relation vector for each direct edge from its child node to parent node, and use it to predict the relation and produce the hidden representation of the higher-level node in the dependency tree. As mentioned above, to address the issues of noisy relation labels, we further incorporate an auto-encoder into RNSCN.

While RNSCN mainly focuses on learning representations using the syntactic interactions among the words, the second component, GRU, aims to compute linear-context interactions. GRU takes the hidden representation of each word computed from RNSCN as inputs and further produces final representation of each word by taking linear contexts into consideration. We describe each component in detail in the following sections.

6.2.1 Recursive Neural Structural Correspondence Network

RNSCN is built upon the dependency tree of each sentence, which is pre-generated from a dependency parser. Specifically, each node in the dependency tree is associated with a word w_n , an input word embedding $\mathbf{x}_n \in \mathbb{R}^D$ and a transformed hidden representation $\mathbf{h}_n \in \mathbb{R}^d$. Each direct edge in the dependency tree is associated with a relation feature vector $\mathbf{r}_{nm} \in \mathbb{R}^d$ and a true relation label vector $\mathbf{y}_{nm}^R \in \mathbb{R}^K$, where K is the total number of dependency relations, n and m denote the indices of the parent word and the child word of the dependency edge, respectively. Based on the dependency tree, the hidden representations are generated in a recursive manner from leaf nodes until reaching the root node. Consider the source-domain review sentence shown in Figure 6.2 as an illustrative example, we first compute hidden representations for the leaf nodes *they* and *good*:

$$\mathbf{h}_1 = \tanh(\mathbf{W}_x \mathbf{x}_1 + \mathbf{b}), \quad \mathbf{h}_3 = \tanh(\mathbf{W}_x \mathbf{x}_3 + \mathbf{b}),$$

where $\mathbf{W}_x \in \mathbb{R}^{d \times D}$ is a transformation matrix for word embeddings. Regarding hidden representation of the non-leaf node *appetizer*, we first generate the relation vector $\mathbf{r}_{43}$ for the dependency edge $\mathbf{x}_4$ (appetizers) $\xrightarrow{\text{amod}} \mathbf{x}_3$ (good) by

$$\mathbf{r}_{43} = \tanh(\mathbf{W}_h \mathbf{h}_3 + \mathbf{W}_x \mathbf{x}_4),$$

where $\mathbf{W}_h \in \mathbb{R}^{d \times d}$ transforms the hidden representation to the relation vector space. We then compute the hidden representation for *appetizer*:

$$\mathbf{h}_4 = \tanh(\mathbf{W}_{\text{amod}} \mathbf{r}_{43} + \mathbf{W}_x \mathbf{x}_4 + \mathbf{b}).$$

Moreover, the relation vector $\mathbf{r}_{43}$ is used for the auxiliary task on relation prediction:

$$\hat{\mathbf{y}}_{43}^R = \text{softmax}(\mathbf{W}_R \mathbf{r}_{43} + \mathbf{b}_R),$$

where $\mathbf{W}_R \in \mathbb{R}^{K \times d}$ is the relation classification matrix. The supervised relation classifier enforces close proximity of $\{\mathbf{r}_{nm}\}$'s within the same dependency relation in the distributed relation vector space. The relation features help to bridge the gap of each word's hidden representation in different domains by incorporating them into the forward computations. In general, the hidden representation $\mathbf{h}_n$ for a non-leaf node is produced through

$$\mathbf{h}_n = \tanh\left(\sum_{m \in \mathcal{M}_n} \mathbf{W}_{R_{nm}} \mathbf{r}_{nm} + \mathbf{W}_x \mathbf{x}_n + \mathbf{b}\right), \quad (\text{Eq. 6.1})$$

where $\mathbf{r}_{nm} = \tanh(\mathbf{W}_h \cdot \mathbf{h}_m + \mathbf{W}_x \cdot \mathbf{x}_n)$, $\mathcal{M}_n$ is the collection of children nodes of w_n , and $\mathbf{W}_{R_{nm}}$ is the relation transformation matrix tied with each relation R_{nm} . The predicted label vector $\hat{\mathbf{y}}_{nm}^R$ for $\mathbf{r}_{nm}$ is computed via

$$\hat{\mathbf{y}}_{nm}^R = \text{softmax}(\mathbf{W}_R \cdot \mathbf{r}_{nm} + \mathbf{b}_R). \quad (\text{Eq. 6.2})$$

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

$$\ell_R = \sum_{k=1}^K -\mathbf{y}_{nm[k]}^R \log \hat{\mathbf{y}}_{nm[k]}^R. \quad (\text{Eq. 6.3})$$

Through the auxiliary task of dependency relation prediction, similar/identical relation enforces participating words close to each other, i.e. the model tries to cluster words with similar syntactic functionalities in different domains. Regarding semantics, we take pre-trained word embeddings as input to the recursive neural network, which captures semantic information that produces similar representations for semantically-similar words. By taking them as input, together with the auxiliary task, our model encodes both semantic and syntactic information.

6.2.2 Reduce Label Noise with Auto-encoders

As discussed in Section 6.1, if each unique relation is an individual class for the auxiliary task, it may be difficult to learn an accurate classifier as the relations generated by the dependency parser can be incorrect. To address the issue of noisy labels, we propose to integrate an autoencoder into RNSCN. Specifically, suppose there is a set of latent groups of relations: $G = \{1, 2, \dots, |G|\}$, where each relation is supposed to belong to only one group. For each relation vector, $\mathbf{r}_{nm}$, an autoencoder is performed before feeding it into the auxiliary classifier in (Eq. 6.2). The goal of the autoencoder is to encode the relation vector to a probability distribution that represents the possibility of assigning this relation to any group. As can be seen in the revised architecture shown in Figure 6.3, each relation vector $\mathbf{r}_{nm}$ is first passed through the autoencoder as follows,

$$p(G_{nm} = i | \mathbf{r}_{nm}) = \frac{\exp(\mathbf{r}_{nm}^\top \mathbf{W}_{enc} \mathbf{g}_i)}{\sum_{j \in G} \exp(\mathbf{r}_{nm}^\top \mathbf{W}_{enc} \mathbf{g}_j)}, \quad (\text{Eq. 6.4})$$

where G_{nm} denotes the inherent relation group for $\mathbf{r}_{nm}$, $\mathbf{g}_i \in \mathbb{R}^d$ represents the feature embedding for group i , and $\mathbf{W}_{enc} \in \mathbb{R}^{d \times d}$ is the encoding matrix that computes bilinear interactions between relation vector $\mathbf{r}_{nm}$ and relation group embedding $\mathbf{g}_i$. Thus, $p(G_{nm} = i | \mathbf{r}_{nm})$ represents the probability of $\mathbf{r}_{nm}$ being mapped to group i . An accumu-

lated relation group embedding is computed as:

$$\mathbf{g}_{nm} = \sum_{i=1}^{|G|} p(G_{nm} = i | \mathbf{r}_{nm}) \mathbf{g}_i. \quad (\text{Eq. 6.5})$$

For decoding, the decoder takes $\mathbf{g}_{nm}$ as input and tries to reconstruct the relation feature input $\mathbf{r}_{nm}$. Moreover, $\mathbf{g}_{nm}$ is also used as the higher-level feature vector for $\mathbf{r}_{nm}$ for predicting the relation label. Therefore, the objective for the auxiliary task in (Eq. 6.3) becomes:

$$\ell_R = \ell_{R_1} + \alpha \ell_{R_2} + \beta \ell_{R_3}, \quad (\text{Eq. 6.6})$$

where

$$\ell_{R_1} = \|\mathbf{r}_{nm} - \mathbf{W}_{dec} \mathbf{g}_{nm}\|_2^2, \quad (\text{Eq. 6.7})$$

$$\ell_{R_2} = \sum_{k=1}^K -\mathbf{y}_{nm[k]}^R \log \hat{\mathbf{y}}_{nm[k]}^R, \quad (\text{Eq. 6.8})$$

$$\ell_{R_3} = \|\mathbf{I} - \bar{\mathbf{G}}^\top \bar{\mathbf{G}}\|_F^2. \quad (\text{Eq. 6.9})$$

Here ℓ_{R_1} is the reconstruction loss with $\mathbf{W}_{dec}$ being the decoding matrix, ℓ_{R_2} follows (Eq. 6.3) with $\hat{\mathbf{y}}_{nm}^R = \text{softmax}(\mathbf{W}_R \mathbf{g}_{nm} + \mathbf{b}_R)$ 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 $\mathbf{g}_i$'s as column vectors. This regularization term enforces orthogonality between $\mathbf{g}_i$ and $\mathbf{g}_j$ for $i \neq j$. 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. We denote RNSCN with auto-encoder by RNSCN⁺.

Figure 6.3: An autoencoder for relation grouping.

6.2.3 Joint Models for Sequence Labeling

RNSCN or RNSCN⁺ focuses on capturing and representing syntactic relations to build a bridge between domains and learn more powerful representations for tokens. However, it ignores the linear-chain correlations among tokens within a sentence, which is important for aspect and opinion terms extraction. Therefore, we propose a joint model, denoted by RNSCN-GRU (RNSCN⁺-GRU), which integrates a GRU-based recurrent neural network on top of RNSCN (RNSCN⁺), i.e., the input for GRU is the hidden representations $\mathbf{h}_n$ learned by RNSCN or RNSCN⁺ for the n -th token in the sentence. For simplicity in presentation, we denote the computation of GRU by using the notation f_{GRU} . To be specific, by taking $\mathbf{h}_n$ as input, the final feature representation $\mathbf{h}'_n$ for each word is obtained through

$$\mathbf{h}'_n = f_{GRU}(\mathbf{h}'_{n-1}, \mathbf{h}_n; \Theta), \quad (\text{Eq. 6.10})$$

where Θ is the collection of the GRU parameters. The final token-level prediction is made through

$$\hat{\mathbf{y}}_n = \text{softmax}(\mathbf{W}_l \cdot \mathbf{h}'_n + \mathbf{b}_l), \quad (\text{Eq. 6.11})$$

where $\mathbf{W}_l \in \mathbb{R}^{5 \times d'}$ transforms a d' -dimensional feature vector to class probabilities (note that we have 5 different classes as defined in Section 3.2).

The second joint model, namely RNSCN-CRF, combines a linear-chain CRF with RNSCN to learn the discriminative mapping from high-level features to labels. The

advantage of CRF is to learn sequential interactions between each pair of adjacent words as well as labels and provide structural outputs. Formally, the joint model aims to output a sequence of labels with maximum conditional probability given its input. Denote by $\mathbf{y}$ a sequence of labels for a sentence and by $\mathbf{H}$ the embedding matrix for each sentence (each column denotes a hidden feature vector of a word in the sentence learned by RNSCN), the inference is computed as:

$$\begin{aligned}\hat{\mathbf{y}} &= \arg \max_{\mathbf{y}} p(\mathbf{y}|\mathbf{H}) \\ &= \arg \max_{\mathbf{y}} \frac{1}{Z(\mathbf{H})} \prod_{c \in C} \exp\langle \mathbf{W}_c, g(\mathbf{H}, \mathbf{y}_c) \rangle\end{aligned}\quad (\text{Eq. 6.12})$$

where C indicates the set of different cliques (unary and pairwise cliques in the context of linear-chain). $\mathbf{W}_c$ is tied for each different $\mathbf{y}_c$, which indicates the labels for clique c . The operator $\langle \cdot, \cdot \rangle$ is the element-wise multiplication, and $g(\cdot)$ produces the concatenation of $\{\mathbf{h}_n\}$'s in a context window of each word. The above two models both consider the sequential interaction of the words within each sentence, but the formalization and training are totally different. We will report the results for both joint models in the experiment section.

6.2.4 Training

Recall that in our cross-domain setting, the labels for terms extraction are only available in the source domain, but the auxiliary relation labels can be automatically produced for both domains via the dependency parser. Besides the source domain labeled data $\mathcal{D}_S = \{(\mathbf{X}_{S_i}, \mathbf{y}_{S_i})\}_{i=1}^{n_S}$, we denote by $\mathcal{D}_R = \{(\mathbf{r}_j, \mathbf{y}_j^R)\}_{j=1}^{n_R}$ the combined source and target domain data with auxiliary relation labels. For training, the total loss consists of token-prediction loss ℓ_T and relation-prediction loss ℓ_R :

$$\mathcal{L} = \sum_{\mathcal{D}_S} \ell_S(\mathbf{y}_{S_i}, \hat{\mathbf{y}}_{S_i}) + \gamma \sum_{\mathcal{D}_R} \ell_R(\mathbf{r}_j, \mathbf{y}_j^R), \quad (\text{Eq. 6.13})$$

where γ is the trade-off parameter, ℓ_S is the cross-entropy loss between the predicted extraction label in (Eq. 6.11) and the ground-truth, and ℓ_R is defined in (Eq. 6.6) for RNSCN⁺ or (Eq. 6.3) for RNSCN. For RNSCN-CRF, the loss becomes the negative log probability of the true label given the corresponding input:

$$\ell_S(\mathbf{y}_{S_i}, \hat{\mathbf{y}}_{S_i}) = -\log(\mathbf{y}_{S_i} | \mathbf{h}_{S_i}). \quad (\text{Eq. 6.14})$$

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 [1], 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. The joint model is trained using back-propagation from the top layer of GRU or CRF to RNSCN until reaching to the input word embeddings in the bottom.

6.3 Experiments

6.3.1 Dataset & Experimental Setup

The data is taken from the benchmark customer reviews in three different domains, namely restaurant, laptop and digital devices. The restaurant domain contains a combination of restaurant reviews from SemEval 2014 task 4 subtask 1 [90] and SemEval 2015 task 12 subtask 1 [89]. The laptop domain consists of laptop reviews from SemEval 2014 task 4 subtask 1. For digital device, we take reviews from [42] containing sentences from 5 digital devices. The statistics for each domain are shown in Table 6.1. In our experiments, we randomly split the data in each domain into training set and testing set with the proportion being 3:1. To obtain more rigorous result, we make three random splits

Dataset	Description	# Sentences	Training	Testing
R	Restaurant	5,841	4,381	1,460
L	Laptop	3,845	2,884	961
D	Device	3,836	2,877	959

Table 6.1: Data statistics with number of sentences.

for each domain and test the learned model on each split. The number of sentences for training and testing after each split is also shown in Table 6.1. Each sentence is labeled with aspect terms and opinion terms.

For each cross-domain task, we conduct both inductive and transductive experiments. Specifically, we train our model only on the training sets from both (labeled) source and (unlabeled) target domains. For testing, the inductive results are obtained using the test data from the target domain, and the transductive results are obtained using the (unlabeled) training data from the target domain. The evaluation metric we used is F1 score. Following the setting from existing work, only exact match could be counted as correct.

For experimental setup, we use Stanford Dependency Parser to generate dependency trees. There are in total 43 different dependency relations, i.e. 43 classes for the auxiliary task. We set the number of latent relation groups as 20. Similar to Section 4.4, the input word features for RNSCN are pre-trained word embeddings using word2vec which is trained on the Yelp dataset and electronics dataset in Amazon reviews. The dimension of word embeddings is 100. Because of the relatively small size of the training data compared with the number of parameters, we firstly pre-train RNSCN for 5 epochs with mini-batch size 30 and rmsprop initialized at 0.01. The joint model of RNSCN⁺-GRU is then trained with rmsprop initialized at 0.001 and mini-batch size 30. The trade-off parameter α , β and γ are set to be 1, 0.001 and 0.1, respectively. The hidden-layer dimension for GRU is 50, and the context window size is 3 for input feature vectors of GRU. For the joint model of RNSCN-CRF, we implement SGD with a decaying learning

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)
RNSCN-CRF	35.26 (1.31)	61.67 (1.35)	32.00 (1.48)	52.81 (1.29)	53.38 (1.49)	67.60 (0.99)	34.63 (1.38)	56.22 (1.10)	48.13 (0.71)	65.06 (0.66)	46.71 (1.16)	61.88 (1.52)
RNSCN-GRU	37.77 (0.45)	62.35 (1.85)	33.02 (0.58)	57.54 (1.27)	53.18 (0.75)	71.44 (0.97)	35.65 (0.77)	60.02 (0.80)	49.62 (0.34)	69.42 (2.27)	45.92 (1.14)	63.85 (1.97)
RNSCN⁺-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)

Table 6.2: Comparisons with different baselines.

rate initialized at 0.02. The context window size is also 3 in this case. Both joint models are trained for 10 epochs. All hyper-parameters are selected based on 3-round random-split validation on terms extraction task in the source domain and relation task over the source and target domain. Specifically, for source-domain validation data, we evaluate the models performance on both terms extraction task and relation prediction task. For target-domain validation data, we evaluate the performance on relation task.

6.3.2 Result & Analysis

We compared our proposed model with several baselines and variants of the proposed model:

- **RNCRF**: A joint model of recursive neural network and CRF proposed in Section 4 for single-domain aspect and opinion terms extraction. We make all the parameters shared across domains for target prediction.
- **RNGRU**: A joint model of RvNN and GRU. The hidden layer of RvNN is taken as input for GRU. We share all the parameters across domains, similar to RNCRF.
- **CrossCRF**: A linear-chain CRF with hand-engineered features that are useful for cross-domain settings [46], e.g., POS tags, dependency relations.

- **RAP**: The Relational Adaptive bootstraPping method proposed by [63] that uses TrAdaBoost to expand lexicons.
- **Hier-Joint**: A recent deep model proposed by [25] that achieves state-of-the-art performance on aspect terms extraction across domains.
- **RNSCN-GRU**: Our proposed joint model integrating auxiliary relation prediction task into RvNN that is further combined with GRU.
- **RNSCN-CRF**: The second proposed model similar to **RNSCN-GRU**, which replace GRU with CRF.
- **RNSCN⁺-GRU**: Our final joint model with auto-encoders to reduce auxiliary label noise.

Note that we do not implement other recent deep adaptation models for comparison [14, 126], because Hier-Joint [25] has already demonstrated better performances than these models. The overall comparison results with the baselines are shown in Table 6.2 with average F1 scores and standard deviations over three random splits. Clearly, the results for aspect terms (AS) transfer are much lower than opinion terms (OP) transfer, which indicate that the aspect terms are usually quite different across domains, whereas the opinion terms could be more common and similar. Hence the ability to adapt the aspect extraction from the source domain to the target domain becomes more crucial. On this behalf, our proposed model shows clear advantage over other baselines for this more difficult transfer problem. Specifically, we achieve 6.77%, 5.88%, 10.55% improvement over the best-performing baselines for aspect extraction in $R \rightarrow L$, $L \rightarrow D$ and $D \rightarrow L$, respectively. By comparing with RNCRF and RNGRU, we show that the structural correspondence network is indeed effective when integrated into RvNN.

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
RNSCN-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
RNSCN-GRU (r)	32.97	50.18	26.21	53.58	35.88	65.73	32.87	57.57	40.03	67.34	40.06	59.18
RNSCN⁺-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
RNSCN ⁺ -GRU (r)	39.27	59.41	33.42	57.24	45.79	69.96	38.21	59.12	45.36	72.84	50.45	68.05

Table 6.3: Comparisons with different variants of the proposed model.

		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	Hier-Joint	33.66	-	33.20	-	48.10	-	31.25	-	47.97	-	34.74	-
	RNSCN ⁺ -GRU*	39.06	-	34.07	-	47.98	-	38.51	-	47.49	-	48.49	-
	RNSCN ⁺	31.60	65.89	24.37	60.01	39.58	71.03	34.40	60.47	41.02	71.23	45.54	69.00
	RNSCN ⁺ -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
IN	Hier-Joint	32.41	-	29.79	-	47.04	-	31.26	-	47.41	-	33.80	-
	RNSCN ⁺ -GRU*	40.34	-	30.75	-	48.69	-	37.40	-	46.49	-	48.50	-
	RNSCN ⁺	30.76	63.65	22.48	59.24	39.54	70.25	35.32	60.00	37.75	70.64	43.72	68.27
	RNSCN ⁺ -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

Table 6.4: Comparisons with different transfer setting.

To show the effect of the integration of the autoencoder, we conduct experiments over different variants of the proposed model. RNSCN-GRU represents the model without autoencoder, which achieves much better F1 scores on most experiments compared with the baselines in Table 6.2. RNSCN⁺-GRU outperforms RNSCN-GRU in almost all experiments. This indicates the autoencoder automatically learns data-dependent groupings, which is able to reduce unnecessary label noise. To further verify that the autoencoder indeed reduces label noise when the parser is inaccurate, we generate new noisy parse trees by replacing some relations within each sentence with a random relation. Specifically, in each source domain, for each relation that connects to any aspect or opinion word, it has 0.5 probability of being replaced by any other relation. In Table 6.3, We denote the model with noisy relations with (r). Obviously, the performance of RNSCN-GRU without an autoencoder significantly deteriorates when the auxiliary labels are very noisy. On the contrary, RNSCN⁺-GRU (r) achieves acceptable results compared to RNSCN⁺-GRU. This proves that the autoencoder 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.

The above comparisons are made using the test data from target domains which are not available during training (i.e., the inductive setting). For more complete comparison, we also conduct experiments in the transductive setting. We pick our best model RNSCN^+ -GRU, and show the effect of different components. To do that, we first remove the sequential structure on top, resulting in RNSCN^+ . Moreover, we create another variant by removing opinion term labels to show the effect of the double propagation between aspect terms and opinion terms. The resulting model is named RNSCN^+ -GRU*. As shown in Table 6.4, we denote by OUT and IN the inductive and transductive setting, respectively. The results shown are the average F1 scores among three splits¹. In general, RNSCN^+ -GRU shows similar performances for both inductive and transductive settings. This indicates the robustness and the ability to learn well when test data is not presented during training. Without opinion labels, RNSCN^+ -GRU* still achieves better results than Hier-Joint most of the time. Its lower performance compared to RNSCN^+ -GRU also indicates that in the cross-domain setting, the dual information between aspects and opinions is beneficial to find appropriate and discriminative relation feature space. Finally, the results for RNSCN^+ by removing GRU are lower than the joint model, which proves the importance of combining syntactic tree structure with sequential modeling.

To qualitatively show the effect of the auxiliary task with autoencoders for clustering syntactically similar words across domains, we provide some case studies on the clustering of words. In Table 6.5, the words within the same group are those that are assigned the same group during encoding of the auto-encoder.

¹We omit standard deviation here due to the limit of space.

Group 1	this, the, their, my, here, it, I, our, not
Group 2	quality, jukebox, maitre-d, sauces, portions, volume, friend, noodles, calamari
Group 3	in, slightly, often, overall, regularly, since, back, much, ago
Group 4	handy, tastier, white, salty, right, vibrant, first, ok
Group 5	get, went, impressed, had, try, said, recommended, call, love
Group 6	is, are, feels, believes, seems, like, will, would

Table 6.5: Case studies on word clustering

6.4.a: On trade-off parameter.

6.4.b: On number of groups.

Figure 6.4: Sensitivity studies for L→D.

In the end, we verify the robustness and capability of the model by conducting sensitivity studies and experiments with varying number of unlabeled target data for training, respectively. Figure 6.4 shows the sensitivity test for L→D, which indicates that changing of the trade-off parameter γ or the number of groups $|G|$ does not affect the model's performance greatly, i.e., less than 1% for aspect extraction and 2% for opinion extraction. This proves that our model is robust and stable against small variations. Figure 6.5 compares the results of RNSCN⁺-GRU with Hier-Joint when increasing the proportion of unlabeled target training data from 0 to 1. Obviously, our model shows steady improvement with the increasing number of unlabeled target data. This pattern proves our model's capability of learning from target domain for adaptation.

6.5.a: F1-aspect on R→L

6.5.b: F1-aspect on D→L

Figure 6.5: F1 vs proportion of unlabeled target data.

6.4 Summary

We propose a novel dependency-tree-based RvNN, namely RNSCN (or RNSCN⁺), for domain adaptation. The model integrates an auxiliary task into representation learning of nodes in the dependency tree. The adaptation takes place in a common relation feature space, which builds the structural correspondences using syntactic relations among the words in each sentence. We further develop a joint model to combine RNSCN/RNSCN⁺ with a sequential labeling model for terms extraction.

Chapter 7

Language Adaptation via Transition-based Adversarial Network

This chapter further extends the task of target terms extraction from a cross-domain perspective to a cross-lingual perspective. The adaptation across languages proves to be more challenging than domain adaptation, because the features from different languages lie in heterogeneous spaces, which is difficult to exploit common knowledge to bridge the gap.

In fine-grained opinion mining, a crucial issue is the difficulty in collecting sufficient labeled data to train a precise classifier, especially for resource-limited languages. Existing cross-lingual approaches mainly focused on coarse-grained sentiment classification. To the best of our knowledge, there is yet no studies on cross-lingual aspect extraction, due to the difficulties in word-level feature adaptation.

Aspect extraction in supervised mono-lingual setting treat the task as a sequence labeling problem that predicts a label for each token. It has also been shown that syntactic relations are crucial to identify target terms [94, 125]. Token-level knowledge transfer has been proposed for cross-domain aspect extraction [63, 25] that are built upon common sentiment lexicons. However, those methods only work for homogeneous space within

the same language. Moreover, token-level knowledge is much more difficult to be transferred across languages, because the segmentation rules can be very different in different languages. For instance, an aspect can be expressed by a single word in one language but multiple words in another language. To take this challenge into consideration, we propose to use transition-based models, which have been used for dependency parsing [28] and named entity recognition [58]. Different from sequence labeling, transition-based models learn a sequence of actions that read words sequentially from a buffer and avoids the separation of phrases. These actions are taken based on the configuration representation at the previous timestamp. To transfer knowledge, we aim to learn a shared space for configurations from different languages.

Recently, adversarial networks [34] have been widely used and shown promising results for domain adaptation [31]. [16] applied the domain adversarial network to cross-lingual sentiment classification which learns a domain discriminator for sentence representations. However, fine-grained adaptation is much more difficult than sharing of sentence representations for coarse document classification across different languages. To make adaptation successful, we propose a transition-based adversarial network, where a generator is used to produce language-invariant configuration features, and a discriminator competes to discriminate the configurations between source and target languages. Since dependency relations have proven to play an important role in identification of aspect terms, and are invariant across different languages when using universal dependency parsers [22], we build partial dependency trees when computing the configuration representations to capture the key components during transfer. We also use these relations as auxiliary labels that helps the adversarial network to learn better representations. Our main contributions are three-fold: 1) We propose a novel transfer strategy that uses sentence configuration as invariant features for cross-lingual word predictions. 2) We integrate syntactic information to assist knowledge transfer across different languages. 3) We conduct extensive experiments to show the effectiveness of our proposed model.

7.1 Problem Definition & Motivation

Problem Definition: The task studied in this chapter involves the identification of explicit aspect terms in each review sentence. Because of the fundamentally different formulations compared to the previous chapters, here we use a different set of definitions. Formally, given a sequence of words $s = (w_1, w_2, \dots, w_n)$, the transition model reads words sequentially and outputs a sequence of actions $\mathbf{y}^q = (y_1^q, y_2^q, \dots, y_m^q)$, resulting in a sequence of configurations $\mathbf{c} = (c_1, c_2, \dots, c_m)$, where m is the number of timestamps. Each $y_t^q \in A$, where $A = \{\text{OUTPUT}, \text{SHIFT}, \text{REDUCE}\}$. Each $c_t = (c_t^o, c_t^c, c_t^b)$ consists of three components that partition the whole sentence: output stack (c_t^o), candidate stack (c_t^c) and buffer (c_t^b), where each component consists of a subsequence of tokens (or empty). The aspect terms are the complete segments in c_t^c when $y_{t+1}^q = \text{REDUCE}$.

Motivation: Transition model has a rich set of actions for parsing, but it can be used with just a few actions for named entity recognition with promising results [58]. With similar idea, we only generate 3 actions, i.e., shift, reduce and output that are useful to produce the aspect chunking. Compared with history-based model with only a stack and a queue, we create 3 components to separately store the processed information (output stack), aspect candidates (candidate stack) and unprocessed information (buffer), so that each components representation indicates different parts that are crucial to decide where to put the next word. The candidate stack emphasizes our main target of identifying aspect terms.

In transition-based models, each configuration c_t represents a partition of the whole sentence at timestamp t . Here, the concept of configuration is our primary motivation for knowledge transfer across different languages. Since the configuration summarizes the current status and decides on the next action, by aligning them from different languages into a shared space, we can transfer the knowledge of how to take actions in the target

language. Compared with token-level representation, the configuration represents the global information including the entire history processed as well as the future ahead. The adaptation within a transition system is more aligned with cognitive modeling: a reader makes incremental actions in the same way regardless of different languages. Moreover, token-level predictions require accurate learning for different positions within word segments, e.g., the prediction for *service staff* should be “BA” (beginning of aspect) for *service* and “IA” (inside of aspect) for *staff*. However, the translation is *personnel de service* in French, which consists of 3 words. When the segments are different, adaptation learning becomes noisy and inaccurate. This could be addressed using transition systems, which use a candidate stack to store and represent a complete aspect segment to be transferred across languages.

It has been shown that universal dependency relations are invariant across different languages. Since [28] showed the effectiveness of a transition model for dependency parsing by predicting the transition or relation at each timestamp according to the current configuration, we take this sequential relation prediction as an auxiliary task to our main task. Moreover, motivated by auxiliary conditioned generative adversarial network (ACGAN) [82], we integrate this auxiliary task into the transition-based adversarial network to make the transfer more coherent with the language-invariant syntactic structure. In addition, we integrate syntactic structure into the representation of each configuration. This helps to make the representations more sensitive to the important words in the sentence. For example, if the stack consists of “We love”, the dependency tree will output the representation for *love*, which is the parent of *We*. This construction is able to attend on the important word *love*.

Figure 7.1: The overall architecture TAN.

7.2 Methodology

We name our proposed model **T**ransition-based **A**dversarial **N**etwork (TAN) in the sequel. Figure 7.1 shows the overall structure which consists of a feature extractor ($\mathcal{F}$) and three classifiers: an action classifier ($\mathcal{Y}$), an auxiliary predictor ($\mathcal{A}$) and a domain discriminator ($\mathcal{D}$). At timestamp t , the input sequence is partitioned into an output stack, a candidate stack and a buffer. Each component computes its own representation through a recurrent neural network (RtNN) in the last layer, which are then integrated into the configuration representation that is fed into all three classifiers. In the following, we first introduce a basic transition-based model in Section 7.2.1, then illustrate the integration of an adversarial component in Section 7.2.2, finally present the final model by considering the syntactic information in Section 7.2.3.

7.2.1 Transition-based Model

The transition-based method can be formalized as a sequence of actions that read words sequentially from a buffer [28]. Specifically, the model consists of an output stack, a candidate stack and a buffer. The output stack stores the processed words, the candidate stack stores incomplete aspect candidates until all the words for an aspect are processed.

Action	Output	Candidate	Buffer	Segment
	[]	[]	[We love the pink pony and atmosphere]	—
OUTPUT	[We]	[]	[love the pink pony and atmosphere]	—
OUTPUT	[We love]	[]	[the pink pony and atmosphere]	—
OUTPUT	[We love the]	[]	[pink pony and atmosphere]	—
SHIFT	[We love the]	[pink]	[pony and atmosphere]	—
SHIFT	[We love the]	[pink pony]	[and atmosphere]	—
REDUCE	[We love the pink pony]	[]	[and atmosphere]	(pink pony)
OUTPUT	[We love the pink pony and]	[]	[atmosphere]	—
SHIFT	[We love the pink pony and]	[atmosphere]	[]	—
REDUCE	[We love the pink pony and atmosphere]	[]	[]	(atmosphere)

Figure 7.2: An example of all the transition actions and states of each configuration of a sentence.

The buffer contains the words that have yet to be processed. There are three possible transition actions: Output, Shift, and Reduce.

- (i) **Output** - Move the word on top of the buffer to the output stack and label it as “None”.
- (ii) **Shift** - Move the word on top of the buffer to the candidate stack.
- (iii) **Reduce** - Move every word in the candidate stack to the output stack and label them as “Aspect”.

The algorithm starts with the buffer containing every word in the sentence. At each configuration, one action is taken, until the algorithm terminates when both the candidate stack and buffer are empty. An illustration of how the actions change the configurations is shown in Figure 7.2. The next action depends on the current configuration, which is computed from the representations of the 3 components: output stack, candidate stack and buffer. We use Gated Recurrent Unit (GRU), one of the RNN architectures to compute the representation of each component. Here a GRU takes the word embedding as the input and produces the output of the last cell to represent the corresponding subsequence, as shown in Figure 7.3 on the left. Note that we take the reverse GRU to compute the buffer representation and use another vector to represent empty stack when the candidate stack is empty.

Figure 7.3: Configuration representation constructed from three recurrent neural networks.

Formally, let $\mathbf{x}_t^O$, $\mathbf{x}_t^C$ and $\mathbf{x}_t^B$ denote the input sequence for output stack, candidate stack and buffer, respectively, at timestamp t . Denote the hidden representations by $\mathbf{h}_t^O$ for output stack, $\mathbf{h}_t^C$ for candidate stack, and $\mathbf{h}_t^B$ for buffer at time stamp t . The final feature representation for the configuration is computed as

$$\mathbf{h}_t = \tanh(\mathbf{W}_h[\mathbf{h}_t^O : \mathbf{h}_t^C : \mathbf{h}_t^B] + \mathbf{b}_h), \quad (\text{Eq. 7.1})$$

where $\mathbf{h}_t^O = f_{GRU}(\mathbf{x}_t^O, \mathbf{h}_0; \Theta)$, $\mathbf{h}_t^C = f_{GRU}(\mathbf{x}_t^C, \mathbf{h}_0; \Theta)$, and $\mathbf{h}_t^B = f_{RGRU}(\mathbf{x}_t^B, \mathbf{h}_0; \Theta)$. Here, the model f_{GRU} produces the last hidden representation of a forward GRU network, and the model f_{RGRU} indicates the reverse GRU network that reads words in the backward direction. We use Θ to denote all the GRU parameters. The action $\mathbf{y}_{t+1}$ to be taken at the next time stamp is obtained via $\mathbf{y}_{t+1}^q = \text{softmax}(\mathbf{W}_q \mathbf{h}_t + \mathbf{b}_q)$.

7.2.2 Transition-based Adversarial Network

To adapt the previous model to another language, we apply domain adversarial network (DAN) to learn domain-invariant features [31]. In our case, the intuition of applying DAN is to learn each configuration representation that is invariant across languages, but discriminative for the action prediction. Indeed, at each state of the transition process, the configuration should share similar characteristics across languages to inform the next

action. To apply DAN in our model setting, we create two classifiers: a transition action predictor $\mathcal{Q}$ and a language discriminator $\mathcal{D}$ to discriminate between configurations from source and target languages.

The corresponding structure of the model is shown on the right part in Figure 7.3. Denote by θ_f , θ_q , and θ_d all the parameters for feature learning, action prediction, and language prediction, respectively. The discriminator generates a probability distribution over languages $P(S|\mathbf{h}_t)$, where $\mathbf{h}_t$ obtained from (Eq. 7.1) is the hidden representation for the whole configuration at time stamp t . Here $P(S = 1|\mathbf{h}_t)$ indicates the probability of $\mathbf{h}_t$ belonging to the source language. The prediction for action label is computed on top of a fully connected layer

$$\mathbf{y}_{t+1}^q = \text{softmax}(\mathbf{W}_q \mathbf{h}_t^q + \mathbf{b}_q), \quad (\text{Eq. 7.2})$$

where $\mathbf{h}_t^q = \tanh(\mathbf{W}_h^q \mathbf{h}_t + \mathbf{b}_h^q)$. Similarly, the language prediction is obtained through

$$\mathbf{y}_t^d = \text{softmax}(\mathbf{W}_d \mathbf{h}_t^d + \mathbf{b}_d), \quad (\text{Eq. 7.3})$$

where $\mathbf{h}_t^d = \tanh(\mathbf{W}_h^d \mathbf{h}_t + \mathbf{b}_h^d)$. The model is trained by minimizing

$$\mathcal{E}_1(\theta_f, \theta_q, \theta_d) = \mathcal{L}_q(\theta_f, \theta_q) - \alpha \mathcal{L}_d(\theta_f, \theta_d) \quad (\text{Eq. 7.4})$$

for feature generator, and by minimizing $\mathcal{L}_d(\theta_f, \theta_d)$ for the language discriminator. Here $\mathcal{L}_q$ and $\mathcal{L}_d$ are the cross-entropy losses for action prediction and language classifier, respectively. The hyper parameter α is used to tune the trade-off between the two losses. The saddle points $\hat{\theta}_f$, $\hat{\theta}_q$, and $\hat{\theta}_d$ can be found as a stationary point of the following updates:

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial \mathcal{L}_q}{\partial \theta_f} - \alpha \frac{\partial \mathcal{L}_d}{\partial \theta_f} \right), \quad (\text{Eq. 7.5})$$

$$\theta_q \leftarrow \theta_q - \mu \frac{\partial \mathcal{L}_q}{\partial \theta_q}, \quad (\text{Eq. 7.6})$$

$$\theta_d \leftarrow \theta_d - \mu \left(\alpha \frac{\partial \mathcal{L}_d}{\partial \theta_d} \right). \quad (\text{Eq. 7.7})$$

Figure 7.4: A configuration with proposed transition-based model.

7.2.3 Transition-based Adversarial Network with Syntactic Relations

The model introduced in the previous section ignores the syntactic relations among the words in each sentence, which is, however, crucial for aspect extraction and bridging the gap between the source language and the target language. To incorporate this information into the deep model, in mono-lingual setting, Chapter 4 proposed to use a dependency-tree-based RvNN that computes the hidden representation of each word as a composition function of its children. Following this idea, a straightforward method is to obtain the hidden representation of each word through RvNN and use it as the input to GRU for the output stack, the candidate stack and the buffer. However, the relation information may have little effect to the action prediction when the dependency path is too long and the model consists of multiple layers. Moreover, the hidden representation for each stack/buffer may contain information from the others due to the dependency tree. This contradicts with the idea that each stack only contains partial information given by its content. To overcome these potential limitations, we propose to construct partial trees in the output stack.

Figure 7.4 presents an example configuration for the proposed model at timestamp t' . For each sentence, we first produce a dependency tree from the parser. At each configuration, we pick all the edges in the tree that connect the words in the output stack and keep the representation for the highest node in each connected component. The rest of the words in the output stack that are not connected are then linked through recurrent edges. As shown in Figure 7.4, the output stack consists of “We love the pink pony and”. According to the dependency tree, *love* is the highest node and is connected to all the other words in the output stack, hence, we only keep the hidden representation of *love* that is recursively computed from all the words below. Formally, given an input sentence with its pre-generated dependency structure, at time stamp t , let $\mathbf{x}_t^O = \{\mathbf{x}_1^O, \dots, \mathbf{x}_{n_t^O}^O\}$, $\mathbf{x}_t^C = \{\mathbf{x}_1^C, \dots, \mathbf{x}_{n_t^C}^C\}$, $\mathbf{x}_t^B = \{\mathbf{x}_1^B, \dots, \mathbf{x}_{n_t^B}^B\}$ represents the list of word embeddings for the output stack, the candidate stack, and the buffer, respectively. For the output stack, we compute the hidden representation h_i^O of each word as:

$$\mathbf{h}_i^O = \tanh \left(\mathbf{W}_v \cdot \mathbf{x}_i^O + \mathbf{b} + \sum_{k \in \mathcal{K}_i^O} \mathbf{W}_{r_{ik}} \cdot \mathbf{h}_k^O \right), \quad (\text{Eq. 7.8})$$

where $\mathcal{K}_i^O$ denotes the set of children of node i in the output stack, r_{ik} denotes the dependency relation between node i and its child node k , and $\mathbf{h}_k^O$ is the hidden vector of the child node k in the output stack. When $\mathcal{K}_i^O = \emptyset$, we ignore the last addition in (Eq. 7.8). To compute the final hidden representation $\mathbf{h}_t^O$ for output stack, we select all the $\mathbf{h}_j^O$'s when the j th node does not have any parent in the output stack or it is the root node in the tree and denote the set of these node indexes as $\mathcal{I}_t^O$:

$$\mathbf{h}_t^O = f_{GRU}(\{\mathbf{h}_j^O\}, h_0; \Theta), \quad (j \in \mathcal{I}_t^O). \quad (\text{Eq. 7.9})$$

The hidden representations for the candidate stack and the buffer are produced by forward recurrent layer and reverse recurrent layer, respectively, as

$$\mathbf{h}_t^C = f_{GRU}(\{\mathbf{h}_i^C\}, \mathbf{h}_0; \Theta), \quad (i \in \{1, \dots, n_t^C\}), \quad (\text{Eq. 7.10})$$

$$\mathbf{h}_t^B = f_{RGRU}(\{\mathbf{h}_i^B\}, \mathbf{h}_0; \Theta), \quad (i \in \{1, \dots, n_t^B\}), \quad (\text{Eq. 7.11})$$

where

$$\mathbf{h}_i^C = \tanh(\mathbf{W}_v \cdot \mathbf{x}_i^C + \mathbf{b}), \text{ \& \ } \mathbf{h}_i^B = \tanh(\mathbf{W}_v \cdot \mathbf{x}_i^B + \mathbf{b}) \quad (\text{Eq. 7.12})$$

We only incorporated the syntactic information in $\mathbf{h}_i^O$ because we assume that only the output stack contains rich processed information. According to the idea of transition, at each timestamp, the buffer and candidate stack contain unprocessed information that needs to be predicted, hence the syntactic information is unknown. Note that we use another embedding to represent the empty stack, which is randomly initialized and trained.

To explicitly make use of the syntactic relations, we integrate an auxiliary task to predict the dependency relation of the word about to be processed in each configuration. Inspired by [28] which applies an auxiliary classifier to help the adversarial training, our auxiliary task to predict the relation of the word on top of the buffer is helpful to decide which action to take in the next timestamp. Moreover, this implicitly conditions the language representations $\mathbf{h}_t^d$ on the relation labels, which helps to better align language features based on syntactic structure. To avoid syntactic inconsistencies across languages, we apply universal dependency parsers and utilize the taxonomy of the universal dependencies¹ to group the relations into more general categories as the auxiliary labels. There are in total 15 categories. The groupings are: $\{nsubj, obj, iobj, nsubjpass, dobj\}$, $\{csubj, ccomp, xcomp\}$, $\{cc, conj\}$, $\{fixed, flat, compound, mwe\}$, $\{expl, obl, vocative, dislocated\}$, $\{advcl\}$, $\{advmod, discourse, neg\}$, $\{aux, cop, mark, auxpass\}$, $\{nmod, appos, nummod\}$, $\{acl\}$, $\{amod\}$, $\{det, clf, case\}$, $\{list, parataxis\}$, $\{root\}$, $\{punct, dep\}$. Denote by θ_a the parameters for auxiliary label prediction. The auxiliary prediction on relations is

$$\mathbf{y}_t^a = \text{softmax}(\mathbf{W}_a \mathbf{h}_t^a + \mathbf{b}_a), \text{ where } \mathbf{h}_t^a = \tanh(\mathbf{W}_h^a \mathbf{h}_t + \mathbf{b}_h^a)$$

By plugging the loss on the auxiliary prediction task into (Eq. 7.4), our final objective becomes

$$\mathcal{E}_2(\theta_f, \theta_q, \theta_a, \theta_d) = \mathcal{L}_q(\theta_f, \theta_q) + \beta \mathcal{L}_a(\theta_f, \theta_a) - \alpha \mathcal{L}_d(\theta_f, \theta_d),$$

¹<http://universaldependencies.org/>

Data	Language	Training	Test	Total
En	English	2,000	676	2,676
Fr	French	1,733	696	2,429
Es	Spanish	2,070	881	2,951

Table 7.1: Dataset statistics showing number of sentences

where $\mathcal{L}_a$ is the auxiliary loss computed with cross-entropy. The updates for θ_q and θ_d are the same as (Eq. 7.6)-(Eq. 7.7). The updates for θ_f and θ_a are:

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial \mathcal{L}_q}{\partial \theta_f} + \beta \frac{\partial \mathcal{L}_a}{\partial \theta_f} - \alpha \frac{\partial \mathcal{L}_d}{\partial \theta_f} \right), \quad (\text{Eq. 7.13})$$

$$\theta_a \leftarrow \theta_a - \mu \left(\beta \frac{\partial \mathcal{L}_a}{\partial \theta_a} \right). \quad (\text{Eq. 7.14})$$

During training, the target language only includes unlabeled training sentences and no pairing of sentences are needed. The training sentences in source language have no correspondences with the training sentences in target language. The language labels are available for all training sentences: 1 for source language, 0 for target language. We combine all training sentences and shuffle them. Each time, a sentence is fed into the model to produce sentence configurations from a common feature extractor. The configuration is fed to the action predictor, auxiliary task and language discriminator for sentence in source language with the ground-truth labels. For target language, only auxiliary task and language discriminator are trained because the target language has no action labels. Given a test sentence, the trained model will produce configuration at each timestamp from feature extractor that is only fed to the action predictor to obtain action labels, as shown in Figure 7.1. These actions generate final aspect segments.

7.3 Experiments

7.3.1 Dataset & Experimental Setup

The datasets for our experiments are taken from SemEval Challenge 2016 task 5. We use the restaurant reviews from English, French and Spanish for the cross-lingual aspect

Models	En→Fr		En→Es		Fr→En		Fr→Es		Es→En		Es→Fr	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Translate-TAN	45.09	40.74	45.85	41.08	39.28	38.74	32.27	34.54	45.94	41.28	41.52	36.38
Translate-CRF	25.23	23.15	28.26	30.10	25.89	26.79	31.55	30.63	32.24	26.66	24.05	20.90
NoAdp	27.71	26.13	27.56	31.31	41.21	38.29	45.43	48.21	37.52	30.39	37.95	37.89
A-RtNN	22.92	20.54	31.11	34.04	29.62	27.11	40.58	40.77	35.49	30.26	34.52	31.02
A-R ² NN	27.92	23.41	28.63	28.65	36.43	33.25	38.55	39.45	40.83	34.16	42.83	37.19
CrossCRF	20.41	16.83	16.17	18.22	21.63	19.02	6.90	6.81	10.13	8.28	12.01	10.24
CL-DSCL	33.67	31.48	44.56	45.01	51.75	47.27	53.23	55.89	50.22	45.90	38.66	34.17
TAN	53.27	50.02	49.38	50.52	55.38	50.30	55.32	57.65	51.99	44.14	51.16	48.78

Table 7.2: Comparisons with different baselines.

extraction. The statistics of the datasets are listed in Table 7.1. In our experiments, we use labeled training data from the source language and unlabeled training data from the target language to train the model. For testing, we conduct both transductive and inductive experiments to test our model on the unlabeled training data and the test data of the target language. The reported results are shown in F1 scores. For aspect extraction, only exact match is counted as correct when dealing with multi-word aspect terms.

Regarding experimental setting, the word embeddings are pre-trained using multi-vec [9] that generates bilingual word embeddings with parallel corpus. We use the parallel corpus from Europarl² that contains 2 million sentences for each language. The dependency trees are generated from universal dependencies of stanford parser³. All the hyper-parameters are chosen based on cross validation. The whole network is trained with SGD using learning rate 0.02. The trade-off parameters α and β are 0.1 and 1, respectively. The size of word embeddings is 100 and the hidden layer size is 50. Each experiment is trained for 20 epochs and the best performance is reported.

7.3.2 Result & Analysis

To the best of our knowledge, there is not yet any work on cross-lingual aspect extraction. To make fair comparisons, we revise some popular models listed in the following to be

²<http://www.statmt.org/europarl/>

³<https://nlp.stanford.edu/software/stanford-dependencies.shtml>

fitted in the cross-lingual setting as our baselines:

- **Translate-TAN**: Translate the target data to source language using google translator and apply TAN trained on the source language to do the prediction.
- **Translate-CRF**: Similar to Translate-TAN, but use linear-chain CRF as the classifier.
- **NoAdp**: Train the model only on source data using proposed network without adaptation and use the model to test on target data.
- **A-RtNN**: Recurrent neural network [66] to produce hidden representation for each word. An adversarial network with a language discriminator is incorporated on top of the hidden representation for each word.
- **A-R²NN**: Similar to Section 4, we use dependency-tree-based recursive neural network with recurrent neural network on top to produce final representation for each word that is then fed into a language discriminator.
- **CrossCRF**: A linear-chain CRF [46] with non-lexical features that are similar across domains. In cross-lingual setting, we use universal dependencies and universal POS tags.
- **CL-DSCL**: A deep model with structural correspondence proposed by [25]. We convert the dependencies in the rules to universal dependencies and use sentiment lexicons for all three languages.

Specifically, for A-RtNN, A-R²NN and CL-DSCL, we combine training sentences from labeled source language and unlabeled target language to train the models. Given a training sentence, A-RtNN and A-R²NN apply RtNN to compute final hidden vector for each word, where a language discriminator is applied to predict the language label (1

for source, 0 for target). If the sentence is from source language, a token classifier is jointly applied to learn “BIO” labeling, given ground-truth token-level labels. CL-DSCL integrates auxiliary classifiers into RtNN. If a training sentence is from source language, both the auxiliary and token-level “BIO” classifier are applied on top of hidden vector of each word. Only auxiliary classifier is applied for sentences in target language. The auxiliary task is to decide whether the word satisfies some rules describing universal dependency relations among aspect and opinion words. CrossCRF only takes training sentences from source language to train. It uses discrete features for each word including non-lexical features that are shared across languages, e.g. universal POS tags, sentiment lexicon, universal dependency relations etc. All the trained models are tested on test sentences for target language to obtain “BIO” label for each token.

The comparison results are shown in Table 7.2. Obviously, the proposed model TAN achieves the state-of-the-art results most of the time with large performance gain. The poor results for Translate-CRF may come from translations that break sequential ordering, especially for informal text. A-RtNN and A-R²NN both apply token-level predictions with adversarial network. Their performances are much worse compared to TAN, even though the syntactic structure is incorporated to bridge the gap between source and target languages (A-R²NN). This proves our assumption that word-level knowledge is hard to be transferred across languages. CL-DSCL modifies the network in [25], which is effective for cross-domain aspect extraction. We convert language-specific dependencies to universal dependencies to facilitate adaptation among different languages. The result shows 18.54%, 5.51% drop for inductive experiments when transfer from English to French and Spanish, respectively. The inferior performances for the cross-domain models indicate the difficulty to build the correspondences with heterogeneous spaces across languages, compared to domain adaptation in the same language. This shows the advantage of our proposed model that learns transferable actions across domains. Regarding model

Models	En→Fr		En→Es		Fr→En		Fr→Es		Es→En		Es→Fr	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
TAN	53.27	50.02	49.38	50.52	55.38	50.30	55.32	57.65	51.99	44.14	51.16	48.78
-hier	50.47	45.73	49.99	50.23	50.70	46.48	50.54	54.07	48.48	42.28	49.81	45.89
-aux	51.30	45.05	47.22	49.00	48.12	45.40	45.17	50.29	44.01	39.05	45.91	42.47
-dan	40.04	36.20	39.87	44.09	47.91	40.40	50.18	52.83	44.44	39.56	42.80	36.04
-ptree	32.62	29.50	28.84	30.94	40.40	35.12	39.73	44.89	32.39	28.75	39.05	35.00
+ftree	48.78	43.28	47.24	48.20	52.68	48.90	50.89	53.50	49.19	43.99	50.65	45.12

Table 7.3: Comparisons with different variants of the proposed model.

complexity, TAN is end-to-end trained directly by backpropagation. The training time is relatively fast, because the parameters only include a few transformation matrices with each dimension less than hundreds, except the word embedding matrix considering the vocabulary size, but each update only involves a few words within a sentence. We use Intel(R) Xeon(R) CPU E5-1650 v2 @ 3.50GHz for experiment. Training 1 epoch with 4000 sentences is 10mins. Testing with 600-800 sentences is within seconds. Training time is similar for A-R²NN. A-RtNN is around 7mins and CL-DSCL is around 15 mins.

On the other hand, we conduct experiments showing the effect of each component of TAN. The results are listed in Table 7.3. The first column indicates the modification of TAN: *-hier* removes the hierarchical grouping of dependency relations in auxiliary task, but use the exact relation as the label; *-aux* removes the auxiliary task; *-dan* removes domain-adversarial network; *-ptree* removes partial tree when computing the output stack, but replace it with GRU and *+ftree* use the complete dependency tree to compute the hidden representation of each word and feed it into a GRU for the representations of output, candidate and buffer. Among all these variants, *-hier* is most comparable with TAN, but still worse than TAN. This shows that grouping of the syntactic relations is more transferable than the relation itself. Indeed, in the hierarchy of universal dependencies, higher level is more general to be shared among different languages. By removing the auxiliary task (*-aux*), the performance drops considerably for the last four experiments. It is even worse when removing DAN (*-dan*). This shows either the auxiliary task or DAN alone is not good enough to capture shared informa-

7.5.a: Es→Fr

7.5.b: En→Es

Figure 7.5: F1 score vs percentage of unlabeled target training data.

tion. By using auxiliary-conditional DAN, we could bridge the gap between source and target language through shared syntactic space. We can also observe the effect of using partial dependency tree to compute the output representation, compared with a simple GRU (*-ptree*) or a complete dependency tree (*+ftree*). This indicates that the recurrent structure is not able to capture syntactic structure of the processed words and may produce unnecessary noise. On the other hand, a complete dependency tree brings excessive knowledge about the words not contained in the stack and make it hard to propagate information when the path is too long.

To show that our model indeed benefits from unlabeled target data during training and learns language-invariant features that are useful for predictions, we conduct experiments to vary the number of unlabeled target data for training. The percentage of the unlabeled target data increases from 0.0 to 1.0. We select two transfer settings, namely Es→Fr and En→Es and compare TAN with 2 baseline methods including CL-DSCL and A-R²NN. The inductive performances are shown in Figure 7.5. Our proposed model shows stable improvements with the increasing number of target training data, while the other 2 baselines do not show clear pattern. This proves that TAN is able to learn from unlabeled target for adaptation.

Figure 7.6: Sensitivity tests for hyper-parameters.

We also conduct sensitivity tests on the hyper-parameters of TAN. As shown in Figure 7.6, we separately vary the trade-off parameter β for Fr→En and the hidden layer dimension of the configuration representation for En→Es. We plot both the transductive and inductive results on training and test data, respectively. The results are overall stable for β and change slowly when changing the dimension of hidden layer from 40 to 100. This proves the robustness and stableness of our proposed model against these variations.

Finally, we provide some case studies to qualitatively show that our proposed model is able to extract aspect terms in target language with different segmentation rules compared to the source language. In Table 7.4, the symbol “——” differentiate the extracted French aspect terms on its left and the corresponding English translation on its right. The left column shows that the same item could be expressed with different number of tokens for different languages. The right column indicates reversal ordering of words for different languages. This manifests that our model is able to transfer knowledge even when different segmentation rules are presented.

carte des vins — wine menu	tarte gourmande — gourmet pie
verre de vin — today's special	pommes sautés — sauteed apples
suggestion du chef — chef's suggestion	desserts maisons — homemade desserts
salade aux noix — walnut salad	menu déjeuner — lunch menu
échine de cochon confite — candied pork loin	cuisine française — French cuisine

Table 7.4: Case studies on extracted aspect terms.

7.4 Summary

In this paper, we propose a novel transition-based deep model for cross-lingual aspect extraction that integrates domain adversarial network with syntactic information. Different from existing methods for fine-grained opinion mining that work on token-level feature representation, our model focuses on learning transferable configuration for the sentence at each timestamp and aims to learn a shared space that is action-sensitive but language-invariant given the syntactic structure.

Chapter 8

Conclusions

In this chapter, we summarize several contributions of this dissertation. We propose a number of deep learning approaches for fine-grained opinion mining, with a special focus on aspect terms and opinion terms extraction. Among all the subtasks within fine-grained opinion mining, the extraction task is most fundamental and crucial that provides necessary information to facilitate further analysis in order to construct structured opinion output. However, the investigation for this task lacks active engagement, with the most probable reason to be the difficulty for exact identification of a text span within non-formal texts. Another difficulty lies in the limited resources, i.e., there are not enough benchmark datasets with sufficient annotations. The most widely used data comes from SemEval challenge that only consists of a few thousand training examples, which makes it hard to train a good model.

To conquer the first limitation, we explore effective deep learning models that could learn a better feature representation for each word within a sentence by exploiting its correlations with other tokens, especially the dual-propagation between aspect terms and opinion terms. These proposed approaches greatly reduce the expensive effort for manually constructing useful features using domain knowledge, and at the same time offer state-of-the-art performances. Moreover, existing methods mostly relied on pre-defined rules, which might be inaccurate for non-formal texts. On the contrary, the proposed deep

models learn the desired patterns in an automatic way. This brings sufficient flexibility to deal with rare cases which are not covered by fixed rules. To conquer the second problem of limited resources, we propose two adaptation methods to extract the target terms in an unsupervised manner. The first method deals with the problem of extraction across domains within the same language. By providing a set of annotated sentences in any source domain, which is easy to find, the model is able to produce desired output for a target domain without any annotations. The second method deals with the problem of extraction across languages. Given labeled data written in a resource-rich language, e.g., English, the model could identify aspect targets written in resource-low languages with acceptable performances. Based on these two general initiatives, we develop several deep models to uncover the complicated interactions among different words, where Chapter 4 and Chapter 5 address the first limitation. Chapter 6 and Chapter 7 address the second limitation.

In Chapter 4, a dependency-tree-based recursive neural network is proposed to model the syntactic interactions within a review sentence. The tree-structured RvNN computes the higher-level representation for each word that incorporates the information of its dependents. This construction allows for propagating the information between aspect terms and opinion terms through their dependency relations without any manual interference. Then we proposed a novel joint model combining the RvNN and a CRF into a unified framework. This model inherits the advantages of RvNN for modeling syntactic interactions and a linear-chain CRF for modeling context interactions. The proposed model is flexible and directly trained through back-propagation, compared to rule-based methods. Despite the promising performance compared to all baseline models, one deficiency could be the reliance towards the pre-generated parse trees. The performance highly depends on the accuracy of the parser, which might be inconsistent for non-formal texts, e.g., tweets. To address this limitation, we propose another deep model based on attention mechanism in Chapter 5.

Chapter 5 adapts the coarse-grained sentence-level attention to word-level, facilitating the exploration of fine-grained interactions between each token and the aspect/opinion prototype. The aspect attention and the opinion attention are coupled in the learning process to double propagates the information between aspect terms and opinion terms automatically without the help of dependency trees. We further investigate a finer-grained problem involving category-specific aspect/opinion terms extraction through multi-task learning strategies. By incorporating a memory network of multi-layer coupled attentions for each task, the framework learns the correlations among different tasks to assist extraction in each aspect category.

Chapter 6 and Chapter 7 propose different unsupervised adaptation methods to transfer knowledge from a source data to a target data. A common intuition is to treat syntactic relations among the words as shared information across different domains or languages to bridge the distribution gap. The idea is that the way each word functions within a sentence should be similar regardless of which domain or language it lies in. Based on this point, the first method propose a recursive neural structural corresponding network that learns a dependency relation feature space that are shared across domains. By incorporating relation learning within each dependency-tree-based RvNN, the learned word representations are clustered according to their syntactic functionalities. Moreover, an auto-encoder is incorporated to cluster different dependency relations in order to reduce the negative effect of inaccurate parsed trees. The second method deals with cross-lingual aspect extraction. Since the words for different languages have no commonalities, their inputs are said to be residing in heterogeneous spaces. In this case, sequential labeling that works on each word is not applicable, because different languages have distinct segmentation rules that are not transferable. Instead, we borrow the idea from transition-based models that produces a sequence of actions to push or pop each word from some stacks. In our case, the actions are used to produces chunking of

an aspect term. The problem becomes to predict the action at each timestamp according to the current sentence configuration. We propose a domain-adversarial transition-based network to transfer the sentence configuration representations across languages. Bearing in mind that different languages share similar syntactic functionalities, we make our network conditioned on the dependency relations to learn more meaningful adaptations.

To sum up, we contribute to a series of novel deep learning approaches by exploiting word-level relationships for more effective extraction. Our models are developed based on two general initiatives: 1) accuracy and flexibility of the model for the identification of a text span. 2) The status of limited resources for complete aspect/opinion annotations that hinders the development of fine-grained opinion mining.

List of Publications

- **Wenya Wang**, Sinno Jialin Pan. Recursive Neural Structural Correspondence Network for Cross-domain Aspect and Opinion Co-Extraction. ACL, 2018.
- **Wenya Wang**, Sinno Jialin Pan. Transition-based Adversarial Network for Cross-lingual Aspect Extraction. IJCAI, 2018.
- **Wenya Wang**, Sinno Jialin Pan, Daniel Dahlmeier, Xiaokui Xiao. Coupled Multi-Layer Attentions for Co-Extraction of Aspect and Opinion Terms. AAAI, 2017.
- **Wenya Wang**, Sinno Jialin Pan, Daniel Dahlmeier, Xiaokui Xiao. Recursive Neural Conditional Random Fields for Aspect-based Sentiment Analysis. EMNLP, 2016.

List of Submissions

- **Wenya Wang**, Sinno Jialin Pan, Daniel Dahlmeier. Memory Networks for Fine-grained Opinion Mining. Artificial Intelligence, 2018.

References

- [1] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.
- [2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, 2008.
- [3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *CoRR abs/1409.0473*, 2014.
- [4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.
- [5] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205, 2007.
- [6] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128, 2006.
- [7] Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. Unsupervised cross-domain word representation learning. In *ACL (1)*, pages 730–740, 2015.
- [8] Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *HLT-ACL*, pages 804–812, 2010.
- [9] Alexandre Brard, Christophe Servan, Olivier Pietquin, and Laurent Besacier. MultiVec: a Multilingual and Multilevel Representation Learning Toolkit for NLP. In *LREC*, 2016.

- [10] Giuseppe Carenini, Raymond T. Ng, and Ed Zwart. K-cap. pages 11–18, 2005.
- [11] Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, 1997.
- [12] Marie catherine De Marneffe and Christopher D. Manning. Stanford typed dependencies manual, 2008.
- [13] Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *NIPS*. 2014.
- [14] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, pages 1627–1634, 2012.
- [15] Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. Recurrent attention network on memory for aspect sentiment analysis. In *EMNLP*, pages 452–461, 2017.
- [16] Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Q. Weinberger, and Claire Cardie. Adversarial deep averaging networks for cross-lingual sentiment classification. *CoRR*, 2016.
- [17] Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. Aspect extraction with automated prior knowledge learning. In *ACL*, pages 347–358, 2014.
- [18] Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.
- [19] Yejin Choi and Claire Cardie. Hierarchical sequential learning for extracting opinions and their attributes. In *ACL*, pages 269–274, 2010.
- [20] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011.

- [21] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *ICML*, pages 193–200, 2007.
- [22] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, 2014.
- [23] Xiaowen Ding, Bing Liu, and Philip S. Yu. A holistic lexicon-based approach to opinion mining. In *WSDM*, 2008.
- [24] Xiaowen Ding, Bing Liu, and Lei Zhang. Entity discovery and assignment for opinion mining applications. In *SIGKDD*, pages 1125–1134, 2009.
- [25] Ying Ding, Jianfei Yu, and Jing Jiang. Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. In *AAAI*, 2017.
- [26] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL*, pages 49–54, 2014.
- [27] Zi-Yi Dou. Capturing user and product information for document level sentiment analysis with deep memory network. In *EMNLP*, pages 521–526, 2017.
- [28] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *ACL*, 2015.
- [29] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.
- [30] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*, pages 153–160, 2009.

- [31] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, 2016.
- [32] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 97–110, 2011.
- [33] Christoph Goller and Andreas Küchler. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, pages 347–352, 1996.
- [34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*. 2014.
- [35] Stephan Gouws, Yoshua Bengio, and Greg Corrado. Bilbowa: Fast bilingual distributed representations without word alignments. In *ICML*, 2015.
- [36] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, pages 1462–1471, 2015.
- [37] Honglei Guo, Huijia Zhu, Zhili Guo, XiaoXun Zhang, and Zhong Su. Product feature categorization with multilevel latent semantic association. In *CIKM*, pages 1087–1096, 2009.
- [38] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. An unsupervised neural attention model for aspect extraction. In *ACL*, pages 388–397, 2017.
- [39] Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. In *ACL*, 2014.
- [40] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

- [41] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [42] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD*, 2004.
- [43] Minlie Huang, Qiao Qian, and Xiaoyan Zhu. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Trans. Inf. Syst.*, 35(3):26:1–26:27, 2017.
- [44] Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *EMNLP*, 2014.
- [45] Mohit Iyyer, Jordan L. Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.
- [46] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *EMNLP*, 2010.
- [47] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *ACL, HLT*, pages 151–160, 2011.
- [48] Wei Jin and Hung Hay Ho. A novel lexicalized hmm-based learning framework for web opinion mining. In *ICML*, pages 465–472, 2009.
- [49] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT*, pages 103–112, 2015.
- [50] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665, 2014.
- [51] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.
- [52] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *ACL*, pages 423–430, 2003.

- [53] Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. Inducing crosslingual distributed representations of words. In *COLING*, 2012.
- [54] Abhishek Kumar and Hal Daum III. Learning task grouping and overlap in multi-task learning. In *ICML*, 2012.
- [55] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, Ishaan Gulrajani, James Bradbury, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, 2016.
- [56] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.
- [57] Himabindu Lakkaraju, Richard Socher, and Christopher D. Manning. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*, 2014.
- [58] Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. Neural architectures for named entity recognition. In *Proc. NAACL-HLT*, 2016.
- [59] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.
- [60] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *EMNLP*, pages 107–117, 2016.
- [61] Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. Structure-aware review mining and summarization. In *COLING*, 2010.
- [62] Fangtao Li, Minlie Huang, and Xiaoyan Zhu. Sentiment analysis with global topics and local dependency. In *AAAI*, 2010.
- [63] Fangtao Li, Sinno Jialin Pan, Ou Jin, Qiang Yang, and Xiaoyan Zhu. Cross-domain co-extraction of sentiment and topic lexicons. In *ACL*, 2012.

- [64] Kang Liu, Liheng Xu, Yang Liu, and Jun Zhao. Opinion target extraction using partially-supervised word alignment model. In *IJCAI*, pages 2134–2140, 2013.
- [65] Kang Liu, Liheng Xu, and Jun Zhao. Opinion target extraction using word-based translation model. In *EMNLP-CoNLL*, pages 1346–1356, 2012.
- [66] Pengfei Liu, Shafiq Joty, and Helen Meng. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*, 2015.
- [67] Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. A logic programming approach to aspect extraction in opinion mining. In *WI*, pages 276–283, 2013.
- [68] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*, pages 912–921, 2015.
- [69] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *WWW*, pages 131–140, 2009.
- [70] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive attention networks for aspect-level sentiment classification. In *IJCAI*, pages 4068–4074, 2017.
- [71] Tengfei Ma and Xiaojun Wan. Opinion target extraction in chinese news comments. In *COLING*, 2010.
- [72] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. pages 1020–1025, 2012.
- [73] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52, 2015.
- [74] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *WWW*, pages 171–180, 2007.

- [75] Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. Cross-lingual mixture model for sentiment classification. In *ACL*, 2012.
- [76] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
- [77] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
- [78] Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, 2013.
- [79] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. 2016.
- [80] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, pages 2204–2212. 2014.
- [81] Rodrigo Moraes, João Francisco Valiati, and Wilson P. Gavião Neto. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Syst. Appl.*, 40(2):621–633, 2013.
- [82] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *ICML*, 2017.
- [83] Nir Ofek, Soujanya Poria, Lior Rokach, Erik Cambria, Amir Hussain, and Asaf Shabtai. Unsupervised commonsense knowledge enrichment for domain-specific sentiment analysis. *Cognitive Computation*, 8(3):467–477, 2016.
- [84] Naoaki Okazaki. Crfsuite: a fast implementation of conditional random fields (CRFs), 2007.
- [85] Junfeng Pan, Gui-Rong Xue, Yong Yu, and Yang Wang. Cross-lingual sentiment classification via bi-view non-negative matrix tri-factorization. In *PAKDD*, 2011.

- [86] Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *WWW*, pages 751–760, 2010.
- [87] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.
- [88] Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphe De Clercq, Vronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeny Kotelnikov, Nuria Bel, Salud Mara Jimnez-Zafra, and Glen Eryiit. SemEval-2016 task 5: Aspect based sentiment analysis. In *SemEval 2016*, 2016.
- [89] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. SemEval-2015 task 12: Aspect based sentiment analysis. In *SemEval 2015*, pages 486–495, 2015.
- [90] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*, pages 27–35, 2014.
- [91] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *EMNLP*, pages 339–346, 2005.
- [92] Peter Prettenhofer and Benno Stein. Cross-language text classification using structural correspondence learning. In *ACL*, 2010.
- [93] Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. Linguistically regularized lstm for sentiment classification. In *ACL*, pages 1679–1689, 2017.
- [94] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.*, 37(1):9–27, 2011.
- [95] Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. Multilinear multitask learning. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1444–1452, 2013.

- [96] Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. A hierarchical model of reviews for aspect-based sentiment analysis. In *EMNLP*, pages 999–1005, 2016.
- [97] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389, 2015.
- [98] Cicero Dos Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *ICML*, volume 32, pages 1818–1826, 2014.
- [99] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *EMNLP*, 2012.
- [100] Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218, 2014.
- [101] Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.
- [102] Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks. pages 1–9, 2010.
- [103] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*, 2011.
- [104] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, 2013.
- [105] Qi Su, Xinying Xu, Honglei Guo, Zhili Guo, Xian Wu, Xiaoxun Zhang, Bin Swen, and Zhong Su. Hidden sentiment association in chinese web opinion mining. In *WWW*, pages 959–968, 2008.

- [106] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.
- [107] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Target-dependent sentiment classification with long short term memory. *CoRR*, abs/1512.01100, 2015.
- [108] Duyu Tang, Bing Qin, and Ting Liu. Learning semantic representations of users and products for document level sentiment classification. In *ACL*, pages 1014–1023, 2015.
- [109] Duyu Tang, Bing Qin, and Ting Liu. Aspect level sentiment classification with deep memory network. *CoRR*, abs/1605.08900, 2016.
- [110] Zhiyang Teng, Duy Tin Vo, and Yue Zhang. Context-sensitive lexicon features for neural sentiment analysis. In *EMNLP*, pages 1629–1638, 2016.
- [111] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *www*, pages 111–120, 2008.
- [112] Ivan Titov and Ryan T. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, pages 308–316, 2008.
- [113] Zhiqiang Toh and Wenting Wang. Dlirec: Aspect term extraction and term polarity classification system. In *SemEval*, pages 235–240, 2014.
- [114] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, pages 173–180, 2003.
- [115] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *NIPS*, pages 2773–2781. 2015.
- [116] Xiaojun Wan. Co-training for cross-lingual sentiment classification. In *ACL-AFNLP*, 2009.
- [117] Hao Wang and Martin Ester. A sentiment-aligned topic model for product aspect rating prediction. In *EMNLP*, 2014.

- [118] Hongning Wang, Yue Lu, and ChengXiang Zhai. Latent aspect rating analysis without aspect keyword supervision. In *KDD*, pages 618–626, 2011.
- [119] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for aspect-level sentiment classification. In *EMNLP*, pages 606–615, 2016.
- [120] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.
- [121] Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. Phrase dependency parsing for opinion mining. In *EMNLP*, pages 1533–1541, 2009.
- [122] Shufeng Xiong, Yue Zhang, Donghong Ji, and Lou Yinxia. Distance metric learning for aspect phrase grouping. In *COLING*, pages 2492–2502, 2016.
- [123] Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. Cached long short-term memory neural networks for document-level sentiment classification. In *EMNLP*, pages 1660–1669, 2016.
- [124] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489, 2016.
- [125] Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. Un-supervised word and dependency path embeddings for aspect term extraction. In *IJCAI*, 2016.
- [126] Yang Yongxin and Hospedales Timothy M. A unified perspective on multi-domain and multi-task learning. In *ICLR*, 2015.
- [127] Jianfei Yu and Jing Jiang. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *EMNLP*, pages 236–246, 2016.
- [128] Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. Aspect ranking: Identifying important product aspects from online consumer reviews. In *ACL*, 2011.

- [129] Shuangfei Zhai and Zhongfei Zhang. Semisupervised autoencoder for sentiment analysis. 2016.
- [130] Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. Grouping product features using semi-supervised learning with soft-constraints. In *COLING*, pages 1272–1280, 2010.
- [131] Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. Clustering product features for opinion mining. In *WSDM*, pages 347–354, 2011.
- [132] Meishan Zhang, Yue Zhang, and Duy-Tin Vo. Gated neural networks for targeted sentiment analysis. In *AAAI*, pages 3087–3093, 2016.
- [133] Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *EMNLP*, pages 56–65, 2010.
- [134] Guangyou Zhou, Tingting He, Jun Zhao, and Wensheng Wu. A subspace learning framework for cross-lingual sentiment classification with partial parallel data. In *IJCAI*, pages 1426–1432, 2015.
- [135] Guangyou Zhou, Zhao Zeng, Jimmy Xiangji Huang, and Tingting He. Transfer learning for cross-lingual sentiment classification with weakly shared deep neural networks. In *SIGIR*, pages 245–254, 2016.
- [136] Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W. Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *AAAI*, 2014.
- [137] Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. Cross-lingual sentiment classification with bilingual document representation learning. In *ACL*, 2016.
- [138] Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Muhua Zhu. Multi-aspect opinion polling from textual reviews. In *CIKM*, pages 1799–1802, 2009.