5-2017

Aspect discovery from product reviews

Ying DING
Singapore Management University, ying.ding.2011@phdis.smu.edu.sg

Aspect Discovery from Product Reviews

YING DING

SINGAPORE MANAGEMENT UNIVERSITY

2017

Aspect Discovery from Product Reviews

by

Ying Ding

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

Dissertation Committee:

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

Hady W. LAUW
Assitant Professor of Information Systems
Singapore Management University

David LO
Associate Professor of Information Systems
Singapore Management University

Yulan HE
Reader
Aston University

Singapore Management University
2017

Aspect Discovery from Product Reviews

Ying Ding

Abstract

With the rapid development of online shopping sites and social media, product reviews are accumulating. These reviews contain information that is valuable to both businesses and customers. To businesses, companies can easily get a large number of feedback of their products, which is difficult to achieve by doing customer survey in the traditional way. To customers, they can know the products they are interested in better by reading reviews, which may be uneasy without online reviews. However, the accumulation has caused consuming all reviews impossible. It is necessary to develop automated techniques to efficiently process them.

One of the most fundamental research problems related to product review analysis is aspect discovery. Aspects are components or attributes of a product or service. Aspect discovery is to find the relevant terms and then cluster them into aspects. As users often evaluate products based on aspects, presenting them with aspect level analysis is very necessary. Meanwhile, aspect discovery works as the basis of many downstream applications, such as aspect level opinion summarization, rating prediction, and product recommendation.

There are three basic steps to go through for aspect discovery. The first one is about defining the aspects we need. In this step, we need to understand and determine what are considered aspects. The second one is about identifying words that are used to describe aspects. This step can help us concentrate on analyzing information that is most relevant to aspect discovery. The third one is about clustering words into aspects. The main goal of this step is to cluster words that are about the same aspect into the same group.

There has been much work trying to do the three basic steps in different ways. However, there still exist some limitations with them. In the first step, most existing

studies assume that they can discover aspects that people use to evaluate products. However, besides aspects, there also exist another type of latent topics in product reviews, which is named "properties" by us. Properties are attributes that are intrinsic to products, which are not suitable to be used to compare different products. In the second step, to identify aspect words, many supervised learning based models have been proposed. While proven to be effective, they require large amounts of training data and turn to be much less useful when applied to data from a different domain. To finish the third step, many extensions of LDA have been proposed for clustering aspect words. Most of them only rely on the co-occurrence statistics of words without considering the semantic meanings of words.

In this dissertation, we try to propose several new models to deal with some remaining problems of existing work:

1. We propose a principled model to separate product properties from aspects and connect both of them with ratings. Our model can effectively do the separation and its output can help us understand users' shopping behaviors and preferences better.

2. We design two Recurrent Neural Network (RNN) based models to incorporate domain independent rules into domain specific supervised learning based neural networks. Our models can improve a lot over some existing strong baselines in the task of cross-domain aspect word identification.

3. We use word embeddings to boost traditional topic modeling of product reviews. The proposed model is more effective in both discovering meaningful aspects and recommending products to users.

4. We propose a model integrating RNN with Neural Topic model (NTM) to jointly identify and cluster aspect words. Our model is able to discover clearer and more coherent aspects. It is also more effective in sentence clustering than the baselines.

Table of Contents

1 Introduction **1**

1.1 Dissertation Structure 7

2 Literature Review **9**

2.1 Opinion Mining of Product Reviews 9

2.2 Traditional Methods for Aspect Discovery 10

2.2.1 Aspect Word Identification 10

2.2.2 Aspect Word Clustering 12

2.2.3 Jointly Identifying and Clustering Aspect Words 14

2.2.4 Jointly Modeling Aspects and Ratings 15

2.3 Neural Networks for Aspect Discovery 18

2.3.1 Word Embeddings 18

2.3.2 Neural Networks for Aspect Word Identification 18

2.3.3 Neural Networks for Aspect Word Clustering 21

3 A Joint Model of Product Properties, Aspects and Ratings for Online Reviews **24**

3.1 Introduction 25

3.2 Model 26

3.2.1 The HFT Model 27

3.2.2 Our Model 28

3.2.3 Parameter Estimation 32

3.3 Experiments . . . 35
3.3.1 Data . . . 35
3.3.2 Experiment Setup . . . 35
3.3.3 Annotation of Ground Truth . . . 36
3.3.4 Discovery of Meaningful Latent Factors . . . 37
3.3.5 Separation of Product Properties and Aspects . . . 38
3.3.6 Rating Prediction . . . 39
3.3.7 Latent Factors Learned by PAR . . . 41
3.4 Discussion . . . 41

4 Recurrent Neural Networks with Auxiliary Labels for Cross-domain Opinion Target Extraction 43
4.1 Introduction . . . 44
4.2 Methodology . . . 46
4.2.1 Notation . . . 46
4.2.2 Overview of Our Method . . . 46
4.2.3 Recurrent Neural Networks for Opinion Target Extraction . 48
4.2.4 Rule-based Auxiliary Labels . . . 49
4.2.5 Two Architectures for Cross-domain Opinion Target Extraction . . . 51
4.2.6 Learning the Parameters . . . 53
4.3 Experiments . . . 54
4.3.1 Models for Comparison . . . 55
4.3.2 Experiment Results . . . 57
4.4 Discussion . . . 60

5 Modelling Product Reviews with Word Vectors 62
5.1 Introduction . . . 62
5.2 Method . . . 64
5.2.1 Problem Formulation and Notation . . . 64

5.2.2 Collaborative Filtering with Word Embedding-based Topic Models . . . 65
5.2.3 Parameter Estimation . . . 67
5.3 Experiment . . . 71
5.3.1 Group Recommendation in Meetup . . . 71
5.3.2 Product Recommendation in Online Review Website . . . 75
5.4 Discussion . . . 80

6 A Neural Network Model for Semi-supervised Review Aspect Identification 81

6.1 Introduction . . . 82
6.2 Method . . . 83
6.2.1 Problem Formulation . . . 83
6.2.2 Model Overview . . . 84
6.2.3 Review Generation Process . . . 85
6.2.4 RNN to Incorporate Context . . . 87
6.2.5 Connections with Topic Models . . . 88
6.2.6 Learning . . . 89
6.3 Experiments . . . 91
6.3.1 Data . . . 92
6.3.2 Aspect Quality . . . 93
6.3.3 Perplexity . . . 95
6.3.4 Sentence Clustering . . . 96
6.4 Discussion . . . 96

7 Dissertation Conclusion and Future Work 98

7.1 Summary . . . 98
7.2 Future Work . . . 100

List of Figures

1.1 A sample review of a restaurant. 1

1.2 Aspect words in a sample review. Aspects are labeled with different colors, green for food, yellow for drink, blue for price and red for service. 3

3.1 Plate notation of (a) HFT and our (b) PAR model. Circles in gray indicate hyperparameters and observations. 27

4.1 A sample sentence and its labels. 46

4.2 Overview of the standard method and our method. 47

4.3 Effect of RNN unit. 58

4.4 Effect of hidden dimensions. 59

5.1 Mean Average Ranking (MAR) for CET, CTR and OCF on Meetup data. 73

5.2 RMSE over topic numbers on two datasets. The left one is *office* dataset, the right one is *patio* dataset. 77

6.1 Perplexities over different numbers of aspects for different models. . 96

6.2 Normalized mutual information. 97

List of Tables

3.1 Statistics of our data sets.*#W/R stands for #Word/Review and Voc stands for vocabulary. 35

3.2 Summary of the ground truth latent factors. 37

3.3 Results for identification of meaningful latent factors 37

3.4 Confusion matrices of PAR for all data sets. *P stands for property and A stands for aspect. 39

3.5 The properties and aspects of SOFT learned by PAR. The second column is the human annotated labels of each latent factor and the third column is the top 10 words of the corresponding latent factor. We can see that most properties are types of software. 40

3.6 Performance in rating prediction. 41

3.7 Ratable aspect topics and their top words identified by PAR and HFT in SOFT and MP3 datasets. The first column shows the methods while the second column and third column is the topic and topic word list respectively. 42

4.1 Rules for detecting opinion targets. `H` represents any word. `W` represents an additional target word to be detected using the expansion rules. 50

4.2 Basic statistics of the datasets. 54

4.3 F1 scores achieved by the various methods we consider. The *Data* column shows the source and the target domains, where *L* stands for laptop, *R* stands for restaurant, *D* stands for device and *S* stands for service. † indicates that the result is statistically significantly better than CRF, mDA, FEMA, Direct-1, Direct-2, Aux and Direct-Aux with $p < 0.01$ based on McNemar's test. As an upper bound, we note that the F1 scores on the four domains *R*, *L*, *D* and *S* when trained on in-domain data are 0.779, 0.766, 0.451 and 0.438, respectively. 57

4.4 Sampled sentences and the targets extracted by different models. We use Service as source domain and restaurant as target domain. The gold standard targets are highlighted in black font. 60

5.1 Notation of our model. 65

5.2 Sampled users, the representative tags of groups they join, the tags of group we should recommend and the percentage ranking of CET and CTR. 74

5.3 Top words of sampled topics learned from Meetup data by CET and CTR. 75

5.4 Dataset statistics, which show number of users, number of items, number of reviews, total number of word types, average number of tokens per review in each column. 76

5.5 RMSE of CET, HFT and RMR. For each dataset, the best result is in bold font. † indicates that CET significantly outperforms RMR at 1% level. ‡ indicates that CET significantly outperforms both RMR and HFT at 1% level. 77

5.6 Topic Coherence 77

5.7 Top words of sampled topics learned by CET, HFT and RMR. . . . 79

6.1 Examples of annotated sentences. Aspect words are highlighted and enclosed with brackets. 84
6.2 Model precision (*MP*) of word intrusion by various models. 93
6.3 Topic coherence. 94
6.4 Sampled learned aspects from the Laptop dataset. 95

Acknowledgements

I want to express my respect and gratitude to many people that have helped me throughout my doctoral study.

First and foremost, I would like to deeply thank my advisor Associate Professor Jing Jiang. I cannot finish my study and earn the degree without her help. As an advisor, she is always nice, responsible and ready to help her students. Her vision and enthusiasm is always inspiring me and helping me stay on the right path. Her encouragement and patience has helped me get through my hard time and keep chasing my dream. My last six years in SMU is a very important period of time that has influenced my personality a lot. Jing was the mentor that guides me both in research and in life. I have learned a lot from her, which will certainly benefit me in my future life.

I would like to thank all other professors in my dissertation committee, Professor Yulan He, Assistant Professor Hady Lauw and Associate Professor David Lo. They have given very generous help and insightful comments to me, which helped me better wrap up our Ph.D study and finish this dissertation. Special thanks to Professor Yulan He, from Aston University in Britain. She has committed a lot of time and energy in reviewing my thesis and giving me a lot of critical suggestions although she was very busy with her own work.

I am deeply grateful to Professor Hwee Hwa Pang, Professor Baihua Zheng and Professor Ee Ping Lim. They have generously helped me many times during my study in SMU. Meanwhile, I would like to thank the administrative team of SMU Ph.D programme, Ms. Chew Hong Ong, Ms. Pei Huan Seow and Ms. Yar Ling

Yeo. They have made my study in SMU very enjoyable.

There are so many friends of mine have provided continuous help to me in the last six years. They have brought me enormous encouragement. However, it is almost impossible to list all their names in this limited space. They are all very important to me and I appreciate their help very much. I want to specially thank my group mates, Swapna Gottipati, Minghui Qiu, Qiming Diao, Cane Leung, Xun Wang, Jianguang Du, Yaliang Li, Lizi Liao, Yang Li, Jianfei Yu, Shuohang Wang, Liangguo Wang, Changlong Yu and Yunshi Lan. I have learned a lot form them. We have had very great time of discussing and sharing our ideas and happinesses.

Lastly, and most importantly, I want to express my great thanks to my wife Dan, my parents and my brother. They are always supporting me and encouraging me to chase my dream. I am proud to be in this family and I dedicate this thesis to them.

To Dan, my parents and my brother.

Chapter 1

Introduction

Online reviews are valuable to both businesses and customers. To businesses, companies can learn what customers like or dislike. They can then adapt their products in the future to satisfy customers' needs. To customers, they can learn the strengths and weaknesses of products based on existing reviews and figure out the products they are mostly satisfied with. Online reviews are becoming more and more important in customers' shopping decision making. Figure 1.1 shows a sample review of a restaurant. By reading this review, we know that the restaurant Saul has good

> I have eaten at Saul, many times, the food is always outrageously good. The duck confit is always amazing and the foie gras terrine with figs was out of this world. But the wine list does have not enough choices. For the price, you cannot eat this well in Manhattan. We went around 9:30 on a Friday and it had died down a bit by then so the service was great!

Figure 1.1: A sample review of a restaurant.

food quality, reasonable price, and nice service. But its wine list does not have enough choices. For a customer who cares a lot about food and price, he/she may try this restaurant after reading this review. However, a customer who values drinks a lot may not go to this place. Meanwhile, after reading this review, the manager of the restaurant may try to enrich their wine list to attract more customers. We can see that online reviews can likely affect both the customers and businesses. They contain valuable information for us to mine and utilize.

One standard way to analyze online reviews is to decompose them into aspects and do aspect-based analysis such as aspect level opinion mining, aspect-based opinion summarization, product recommendation and so on. Aspects are components or attributes of a product or service. For example, "screen" is an aspect of iPhone 7, "service" is an aspect of a restaurant. It is obvious to see that the sample review in Figure 1.1 covers some aspects of restaurants like *food*, *drink*, *service*, and *price*. There are also some other aspects like ambiance, location, and parking that are not included in the review in Figure 1.1. People care about these aspects and also evaluate restaurants based on these aspects.

The goal of aspect discovery from product reviews is to identify aspect words and group them into aspects. For example, "service," "serve," and "waiter" are all words referring to the *service* aspect. So we should group them together to represent the *service* aspect. The sentiment, opinions, and descriptions about these words can then be grouped into the same aspect. Only after this, aspect-based analysis can be accomplished. However, in product reviews, there are no labels about aspects and the volume can be very high, which makes it impossible to manually identify aspects. So we need to develop algorithms to do automated aspect discovery.

Aspect discovery is important to us as it can benefit a lot of downstream applications. Figure 1.2 shows the latent aspects in the review from Figure 1.1 that should be discovered. Aspect words are grouped and shown in different colors. Based on this, we can summarize that the user feels the food of Saul is "always outrageously good," the service is "great" and the wine list "does not have enough choices." Various applications can be built based on this. We can do rating prediction at aspect level to get an evaluation over each aspect. We can summarize users' opinions to understand the detailed comments on each aspect. We can also recommend products to users and present the reasons in text to increase the quality of recommendation. These applications may help us improve users' shopping experiences and satisfactions.

There are three basic steps in aspect discovery from product reviews. The first

I have eaten at Saul, many times, the **food** is always outrageously good. The **duck confit** is always amazing and the **foie gras terrine with figs** was out of this world. But the **wine list** does have not enough choices. For the **price**, you cannot eat this well in Manhattan. We went around 9:30 on a Friday and it had died down a bit by then so the **service** was great!

Figure 1.2: Aspect words in a sample review. Aspects are labeled with different colors, green for food, yellow for drink, blue for price and red for service.

step is to answer *"what are considered aspects,"* which aims at determining the aspects we try to discover. Before doing aspect discovery, we need to first understand and determine what are considered aspects. Generally, people do not think too much about this problem. Most previous work does not explicitly study this problem as they assume that topics learned by topic modeling are naturally aspects they need. In this thesis, we will take a closer look at this problem. The second step is to *identify aspect words*, which is about identifying words that are referring to aspects in product reviews. For example, in the sentence "The food is always outrageously good," the word "food" is the aspect word that should be identified. By doing aspect word identification, we can concentrate on analyzing information that is most relevant to aspect discovery. The third step is about *"how to group aspect words into aspects."* The main concern in this problem is designing models to effectively cluster words that are about the same aspect together. For example, "service," "serve," and "waiter" should be clustered into one aspect as they are all about aspect *service*. This problem can also be considered as a normalization of aspect words. While these three steps are closely connected, some work skips the first two steps and directly studies step three [44, 48, 95]. There is also some work focusing on step two [33, 51, 77] only.

Existing work tries to solve these problems separately or jointly but there are still limitations of them. For step one, to decide "what are considered aspects," some previous work assumes that all aspects they get are rateable aspects that users use to evaluate a product. However, there is more information to discover than this. Besides aspects, there are also product properties, which are specific to products.

While an aspect is a criterion that can be used to compare all or most products in the same category, a property is a feature or attribute that a product either possesses or does not possess such as the brand, location or genre. Most previous work aims at discovering aspects that users use to evaluate products by using topic models. However, they either miss properties or fail to separate properties from aspects. It is important for us to discover both aspects and product properties and separate them as the results can help us better understand users' shopping behaviors and preferences.

For step two, to *identify aspect word*, different strategies have been proposed. They can be categorized into the following categories:

- Frequency and rule-based extraction. When people write reviews about the product, their vocabulary may converge and those nouns that are frequently talked about are usually aspect words that are important to analyze [50]. Most frequency-based aspect word identification algorithms are based on this assumption. Rule-based techniques extract aspect words according to their relations with opinion expressions [77]. In product reviews, aspect words usually come together with opinion expressions and there exist some frequent syntactic relations between them, which can presumably help us with aspect word extraction. Frequency and rules are often jointly used.

- Supervised model based extraction. Identifying aspect words can also be treated as a classification problem. Most previous work adopted sequence labeling techniques as they can effectively model lexical and syntactic patterns [50].

While these techniques have been shown to be effective, there are still some flaws with them. Although frequency and rule-based extraction can achieve reasonable performance, it is difficult to get further improvement as rules cannot cover implicit opinion expressions well. Supervised extraction requires a large amount of annotated data and the trained model is domain specific. It is presumably useful

to combine domain independent rules and domain dependent supervised model to build a model that can effectively identify aspect words across domains. But few work has done this.

For step three, many extensions of Latent Dirichlet Allocation (LDA) [2] have been proposed to cluster words into aspects. Some try to discover more meaningful aspects by developing more complicated models [87] or leveraging external supervisions [11]. Some focus on discovering both aspects and sentiments to have a more comprehensive analysis [48]. There is also some work trying to connect aspects with ratings by developing new principled models [57]. However, most of these studies only rely on co-occurrence statistics of lexical terms. They do not take the semantic meaning of words into consideration. Word embeddings, which emerge in recent years, have been widely used in various NLP tasks as they can encode the semantic and syntactic information of words. They should also be useful in boosting topic models if we can incorporate them well. This is especially important to product reviews as the length of product reviews varies and some can be very short and the words used in reviews are usually very diverse. Traditional topic models may not work well in this scenario.

Previous work has also tried to jointly model aspect word identification and aspect word clustering. Most of them do this is by injecting distant supervisions into topic models. One type of distant supervision is prior knowledge defined by human or mined from external resources. In [32], the authors use some sets of seed words to represent the topics users are interested in and design three different topic models to incorporate these seed words. Domain knowledge mined from external/internal resources has also been used in some other work [10, 11, 12]. Another type of distant supervision is a supervised classifier trained on an external dataset. The output of the classifier can be used to guide the generative process of documents, which will influence the learned aspects in the end. Zhao et al. [103] propose a topic model with a Maximum Entropy classifier as the distant supervision. The Maximum Entropy classifier is trained to separate aspect words from opinion words and

background words. Existing work show that jointly identifying and clustering aspect words can help us learn aspects of higher quality. However, most of these work still relies on traditional topic models and treats a document as a bag of words. They fail to consider lexical semantic meanings of words and they do not incorporate the state-of-the-art techniques (e.g. RNN) for aspect word identification.

As we can see, while there has been some work dealing with each of the three basic problems in aspect discovery from product reviews, there are still some unsolved problems. In this thesis, we point out some challenges in each of the three steps and try to address them by developing some principled models.

To separate properties from aspects, we propose a model [19] that extends an existing work that jointly models ratings and text [57] in Chapter 3. While this existing work [57] works well in using review text to improve product recommendation and providing corresponding explanations, it often mixes product properties with aspects. As stated before, we define "aspects" to be criteria that can be used to compare all or most products in the same category. We also define "product properties" to be latent factors that can explain user preferences but are intrinsic to only certain products. Separating these two types of information in product reviews can help us better understand users' preferences and products' qualities. As they can both influence a user's rating on an item, we link them with the ratings associated with reviews and develop a principled model to jointly model aspects, properties, and ratings. Experiments over three datasets from different domains show that our model can effectively separate properties from aspects.

To combine the power of domain independent rules and domain dependent supervised models together, we design two models [23] to incorporate rules into supervised learning based RNN in Chapter 4. While syntactic rules can learn domain independent knowledge, labeled training data can tune RNN to be able to model implicit patterns. A combination of these two types of knowledge can help us learn features that may lead to better performance in cross-domain aspect word identification. We first use domain independent rules to generate auxiliary labels of the input

data. Then, we design two RNN-based models that can predict both true labels and auxiliary labels. Experiments demonstrate that our models are effective in cross-domain aspect word identification as they can outperform some strong baselines with a large margin.

To model semantic meaning of words in topic models, we propose to model online reviews by making use of word embeddings in Chapter 5 [20]. Our model can seamlessly integrate word vectors into an LDA-based topic model. Meanwhile, we also jointly model ratings, latent factors, topics, and word embedding vectors simultaneously. Incorporating word embeddings can make the combination of content and collaborative filtering more effective. Experiments based on real-world datasets show that our model is able to model latent topics in product reviews better and it is also effective in product recommendation.

To jointly model aspect word identification and aspect word clustering, we propose to integrate RNN with Neural Topic Model (NTM) in Chapter 6 [22]. The NTM we use can model the semantic meanings of words by using word vectors to represent them. An RNN trained on a small scale external dataset is used to guide the generation of product reviews. Experiments on two datasets from different domains demonstrate that our model is able to learn aspects of high quality.

1.1 Dissertation Structure

In the remaining part of this dissertation, we will first review related literature in Chapter 2. Then we will present our work on jointly modeling aspects, properties, and ratings in Chapter 3. In this work, we designed a principled probabilistic model that can separate rateable aspects from product properties and link both of them to ratings, which help us better understand users' rating behaviors. Chapter 4 covers our work of combining domain independent rules and RNN for cross-domain aspect word identification. We propose several RNN-based neural networks to utilize auxiliary labels generated by domain independent rules. In Chapter 5, we will intro-

duce our work of using word vectors to model product reviews. Our work models the semantic meaning of words in topic models by incorporating representational vectors of words into a probabilistic graphic model. Next, we present our work of using an integration of NTM and RNN to jointly identify and cluster aspect words in Chapter 6. With a small scale labeled dataset, our model is able to discover aspects of higher quality. In the end, we conclude this thesis and introduce some potential future work in Chapter 7.

Chapter 2

Literature Review

Because of the importance and accumulation of online product reviews, people haven done various analysis of them. Among these studies, aspect discovery is one of the most popular directions and researchers have tried to address it from different angles. Among the three basic steps, Step 2 and Step 3 have attracted most of the attentions. However, most previous work skips Step 1 as they assume that topics learned by topic models are naturally aspects. In this section, we will first give an overview of research on opinion mining of product reviews as this thesis falls in this category. We then review previous work on aspect word identification, aspect word clustering and joint modeling of both, from which the inspiration of this thesis comes. Recently, neural networks have also been applied to aspect discovery and shown to be amazingly effective. So we also review work related to this direction in the last subsection.

2.1 Opinion Mining of Product Reviews

Opinion mining of product reviews has been a popular research problem for decades. Researchers have studied different sub-problems and achieved significant progress.

Sentiment classification is one of the earliest directions that people start to look

at [58, 72]. It is typically modeled as a classification problem and various algorithms have been proposed for it. As online reviews are all written by users, there may exist spam reviews or helpless reviews. To detect spam reviews and identify helpful ones, people also study problems like spam review detection [37, 45, 47] and review helpfulness score prediction [16, 54, 69]. As the volume of online reviews is high, it is important to study how to help users better consume product reviews. To do this, researchers have studied opinion retrieval and search [30, 100]. Various techniques have been proposed for opinion summarization [21, 55, 67], which aims at generating a summary of opinions for users to quickly digest.

Aspect discovery is a fundamental problem that can be utilized in many of the problems mentioned above. This thesis focuses on aspect discovery as it is fundamentally important and improvement in it can benefit many other problems.

2.2 Traditional Methods for Aspect Discovery

Existing work of aspect discovery from product reviews is mainly composed of two lines of work: (1) aspect word identification, and (2) aspect word clustering. In the following sections, we will review these two lines of research separately.

2.2.1 Aspect Word Identification

There are mainly two lines of work doing aspect word identification: (1) frequency and rule based methods, and (2) supervised learning based methods. Frequency and rule based methods mainly rely on constraints and rules designed by human. Supervised learning based methods need to utilize machine learning techniques and human crafted features.

Frequency and Rule based Methods

Frequency and syntactic rules are usually jointly used to identify aspect word [27, 35, 77, 105]. Words used in reviews of the same type of product usually converge

and some frequent words are normally important aspect words that people use to describe components or attributes of products. This phenomenon triggers researchers to use frequency and other constraints to identify aspect words. As aspect words are nouns or noun phrases most of the time, POS tagging is usually applied first to find candidates. Since opinion expressions usually go together with aspect words, it is presumably useful to use syntactic relations to identify aspect words. Based on this idea, researchers have developed various models for aspect word identification.

Frequency and rule based methods are usually domain general. To utilize this property, we propose to generate auxiliary labels based on syntactic rules and incorporate the auxiliary labels into the training of neural networks in Chapter 4.

Supervised Learning based Methods

Aspect word identification can also be treated as an information extraction problem. Various supervised algorithms have been proposed for it. Among them, sequence labeling models, such as Hidden Markov Model (HMM) [78] and Conditional Random Field (CRF) [43] are the most dominant ones. Both lexical features and POS tags are used in an HMM model to jointly identify aspect words and their sentiment polarities in [36]. Dependency relations are also used as features in a CRF model to do single- and cross-domain aspect word identification [33]. Li et. al. [44] propose to use Skip-chain CRF and Tree CRF to model the conjunction structure and syntactic tree structure for aspect word identification. In [94], HMM is used to model the generation of web content. A probabilistic graphical model is integrated with HMM to extract and cluster aspects.

In recent years, the popularity of deep learning has attracted much attention on neural networks. Neural networks have also be applied to aspect word identification and shown to be effective. An overview of these techniques will be given in Chapter 2.3.

Identifying aspect words is basically a classification problem. As there are far more negative instances (words or phrases that are not describing aspects) than pos-

itive instances, we should not use accuracy, which is widely used in many classification problems, as the evaluation metric. Precision, recall and F1 score of the positive class are metrics commonly used in previous work [33, 36, 94]. If a phrase or a word identified by a model matches with the ground truth, it will be counted as a true positive case. Both exact matching and overlapping have been defined as a match, but exact matching are now more commonly used. Then, precision P, recall R and F1 score F can be computed as:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F = \frac{2PR}{P + R}. \tag{2.1}$$

Here, TP, FP and FN refer to the number of true positive cases, false positive cases and false negative cases respectively. In Chapter 4, we also use these metrics for evaluation when comparing with the baselines.

While supervised learning based methods are effective in identifying aspect words, they require a large amount of annotated training data and the models are usually domain specific. Our neural network based sequence labeling models in Chapter 4 are proposed to deal with this limitation.

2.2.2 Aspect Word Clustering

The purpose of aspect word clustering is to group words belonging to the same aspect together. Topic modeling has become the most dominant technique for this problem. There are mainly three lines of topic modeling based aspect word clustering methods:

- Simultaneously clustering aspects and sentiments. As aspect words are usually target words in opinions while sentiment words are usually used as opinion expressions, it is natural to model the dependency between them and cluster them at the same time. The authors of [48] first do this by assuming each topic is associated with a sentiment distribution. Simple prior information based on sentiment lexicons is also used in this work. The authors

of [5, 38] extend this work by assuming that all words in a sentence are from the same topic. Local dependency is further modeled in [44] by using an Hidden Markov Model (HMM). A similar idea is also applied in [82], but this work deals with review snippets instead of raw review documents and it can separate sentiment words from aspect words. Moghaddam and her co-authors design a new topic model to jointly model aspects and sentiments [61]. As a follow up work, they summarize and compare several alternative ways to design topic models for aspect based opinion mining [62].

- Using external resources to learn better aspects. To further improve the performance of unsupervised topic models, some distant supervisions based on domain knowledge or prior information have been incorporated. A topic model that can take rules as prior knowledge is proposed in [12]. Based on this work, the authors further proposed a model to exploit prior knowledge of lexical relations in dictionaries in [11]. Automated extraction of knowledge to incorporate into topic models is proposed in [10].

To evaluate the quality of discovered topics, researchers have developed different evaluation metrics. Perplexity is a metric measuring the likelihood of generating a left out testing dataset based on a trained model. It is first used in LDA [2] and then adopted by many other work on clustering aspect words in product reviews [61, 62, 63]. However, this metric only measures the generalization ability of topic models. It does not evaluate the quality of discovered topics directly. Topic coherence [60] is a metric to measure how coherent the learned topics are, which is widely used in existing work [11, 12]. Based on the intuition that words co-occur more frequently are more likely to form a topic, it uses statistics of word co-occurrences of the top words in all discovered topics to measure their qualities. Human evaluation is also an important way as only topics that look meaningful to human can be regarded as topics of high quality. Word intrusion [7, 31] is a popular metric based on human evaluation. It first mixed a certain number of top words

from a topic with a top word from another topic and then ask human annotators to pick out the word that they think not belonging to the same topic with the others. It is intuitive that the easier human annotators can correctly pick out the intruded words, the better a model is at learning meaningful aspects. Another alternative way to measure topic quality is to ask human to check if a learned topic is meaningful and also pick out the words that are highly related to the meaningful topics [11, 12]. The underlying idea is quite similar to word intrusion, but it requires more human efforts. To comprehensively compare our model with previous work on aspect quality, we use topic coherence in Section 5.3 and 6.3 for evaluation. Meanwhile, we also test our proposed models on perplexity and word intrusion in Section 6.3.

While these studies haven been proven to be useful in clustering aspect words and some other tasks, they only rely on co-occurrences of words to discover latent topics. Semantic meanings of words are missed. In Chapter 5 and Chapter 6, we propose a model that can cluster words into aspects based on both statistics of training data and semantic meanings of words. Another problem with these studies is that properties are not separated from aspects. Some work even fails to discover rateable latent aspect [57]. We try to tackle this problem in Chapter 3 by designing a new topic model that separate properties from aspects with the help of user and product identity and ratings.

2.2.3 Jointly Identifying and Clustering Aspect Words

Identifying and clustering aspect words can be jointly modeled. To this end, various models have been proposed. Distant supervision based on prior knowledge is one of the most popular ways. In [32], the authors try to guide topic models to learn topics of specific interest to a user. They provide sets of seed words the user believes are representative of the topics he/she is interested in and design 3 different extensions of LDA to incorporate these seed words. To further improve the performance of unsupervised topic models, more advanced models [10, 11, 12] have also been

developed to incorporate distant supervision based on domain knowledge or prior information. Supervised learning models can also be used to guide the discovery of aspects. Zhao et. al. [103] use a Maximum Entropy classifier to separate aspect words from opinion words and background words. A topic model is integrated with this classifier to jointly cluster aspect words. These work rely only co-occurrence of words to learn topics without considering their semantic meanings and do not leverage the state-of-the-art deep learning techniques for aspect word identification. In Chapter 6, we propose a new model to overcome these shortcomings.

2.2.4 Jointly Modeling Aspects and Ratings

The popularity of online shopping and online content or service providers have made it easy for people to rate products and services. Users' ratings are important information that can help us learn user preferences and product/service qualities. Compared with reviews, ratings are almost effortless for users to give, which also means they are easier to be collected. Jointly modeling both reviews and ratings may bring us additional benefits. There has been several work on this topic and our work presented in Chapter 3 and Chapter 5 also fall into this category.

Most of existing work on this topic is based on Collaborative Topic Regression [90], which first combines topic models with matrix factorization. It is designed to recommend papers to users by using topic models to model the content of papers and using matrix factorization to model users' adoptions of papers. In this model, each paper is treated as a document, which is generated following LDA. While the topic distribution of paper i is θ_i, its latent factor v_i is sampled from a normal distribution with θ_i as the mean:

$$v_i \sim \mathcal{N}(\theta_i, \sigma). \tag{2.2}$$

By connecting θ_i with v_i, we can regularize papers' latent factors to be close to their topic distributions. Then papers with similar content will also have similar latent

factors. It is assumed that each user u also have a latent factor v_u, which is sampled from a multivariate Gaussian distribution. The inner product of v_i and v_u is used to approximate user u adoption of item i:

$$\hat{r}_{ui} = v_u^\intercal v_i. \tag{2.3}$$

All parameters are jointly learned by using gradient descent algorithm. Some other work [57, 49] differ from CTR by using different hypothesis when modeling the connection between topic distributions and latent factors. In [57], the authors try to model product reviews for personalized rating prediction. They group all reviews of the same product together to form a document for topic modeling. While their framework is very similar to CTR, they assume that the product latent factor v_i is connected to the topic distribution θ_i of its reviews by an exponential transformation:

$$\theta_{ij} = \frac{\exp(\kappa v_{ij})}{\sum_{j'} \exp(\kappa v_{ij'})} \tag{2.4}$$

Ling et al. [49] propose another extension of CTR. They assume that to each user, there is one Gaussian distribution for each topic, which models his preference on that topic. User u's rating on item i is sampled from a Gaussian mixture distribution with the topic distribution θ_i, which is learned from all reviews about i, as the mixture probability.

Most previous work only connect latent factors with topic distributions without considering sentiments and opinions. However, it is intuitive that opinions in a product review have a strong correlations with its rating. Based on this intuition, several models are proposed [18, 95]. They both connect sentiments with ratings by assuming that an aspect specific sentiment distribution is determined by the corresponding aspect specific rating. When generating a word in a review, we should either sample it from an aspect word distribution or a aspect-sentiment word distribution. Besides, some researchers [1] have also taken product review helpfulness

score into consideration when jointly modeling ratings and reviews. The core idea of this work is to assign weights to training instances according to their helpfulness score.

Different from work focusing on clustering aspect words, work that jointly model ratings and reviews mainly care about their performances in recommending products to users. Root Mean Squared Error RMSE is one of the most widely used evaluation metric for this [1, 18, 49, 57, 95], which measures the difference between true ratings and the predicted ones by a model. We also use this metric to evaluate our model in the experiments in Section 3.3 and Section 5.3. More detailed explanations of this metric can also be found in these two sections.

While these work has achieved some improvements in recommending products, they still have quite a few problems to be solved. Firstly, empirical experiments show that the topics discovered by some of these models are intrinsic properties of products instead of rateable aspects [57]. This indicates that these models cannot fully utilize the information hidden in product reviews. To overcome this problem, we propose a new principled model to separate aspects and properties in Chapter 3, which also jointly models ratings and reviews simultaneously. Secondly, most existing work still rely on topic models to model the content of product reviews. Semantic meanings of words are not considered. However, traditional topic models may not work well on product reviews as reviews can have a larger diversity in terms of document length and word usage compared with news articles. Taking semantic similarities between words into consideration can help us overcome this problem. Towards this end, we propose a model that can leverage word embedding vectors, which encode the semantic and syntactic meanings of words, in Chapter 5.

2.3 Neural Networks for Aspect Discovery

2.3.1 Word Embeddings

Word embedding is a recently proposed technique inspired by advances in deep neural networks [59, 73]. Based on the learning from large corpora, it can represent words with numerical vectors that encode their syntactic and semantic meanings. The similarity between vectors of words that are semantically or syntactically similar will be high. It has been used in different applications such as information retrieval [14] and text summarization [42].

Traditional word embedding models assume that one word owns only on dense vector. This is problematic as the semantic meanings and syntactic roles of a word may vary in different context. Reisinger and Mooney [79] propose a model for multi-prototype word vectors by first cluster word contexts and then learn word vectors for each context. The similar idea is also used in [29]. The authors of [53] tackle this problem from another angle by learn different word vectors under different topics.

2.3.2 Neural Networks for Aspect Word Identification

Neural networks have been shown to be effective in aspect words identification. The authors of [51] use standard RNN models [24, 39] and LSTM [26] for this task. CRF is extended to incorporate continuous features and nonlinear functions in [101]. The authors aim at tackling two different problems, aspect word identification and targeted sentiment classification at the same time. Three different models are proposed: a pipeline model, a collapsed model and a joint model. Discrete features that are commonly used in traditional sequence labeling problems are also combined with neural features. Experiments show that the combination can improve the performance. The pipeline model is proved to be the best based on one English Tweet dataset and one Spanish Tweet dataset. While [51] and [101]

both use neural networks to model aspect word extraction problem, they only utilize the linear chain structure of a sentence. Long range dependency and syntactic structures are not considered in these models. In [98], the authors propose to model syntactic dependencies by using dependency path embeddings as input. Similar to the training of word embeddings, dependency path embeddings are trained based on a large corpus. Path embeddings encode the information of dependency relations as continuous vectors, which can be later used as features for other models. Both word embeddings and dependency path embeddings are then discretized and used as input features to a CRF model. The dependency tree structure can also be directly used for aspect word identification [93]. Recursive Neural Network is used to learn the representation of each position. On top of this, a CRF model is further used to model the sequence dependency of labels. Although neural networks are shown to be effective in aspect word identification, it may turn to be less useful if the testing data is from a different domain than the training data. This is actually a common problem with all supervised based aspect identification algorithms.

Our model in Chapter 4 is based on RNN. RNN is a special class of artificial neural networks with cyclical connections. They take a sequence as input and create internal memory at each position, which is used to learn the representation of the input. They can be used in various NLP tasks, such as POS tagging [74], information extraction [51], text classification [85] and many others. Different types of RNN units have been developed, which differ in how internal memory is generated. The two most basic RNN models are Elman-RNN [24] and Jordan-RNN [39]. Assume that $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$ is an input sequence where each $\mathbf{x}_t$ in it is a vector. An RNN aims at learning a hidden representational vector $\mathbf{h}_t$ for each position t that supposedly encodes all the input from the beginning of the sequence up to position t. $\mathbf{h}_t$ is defined as:

$$\mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

Based on this hidden representation, we can further get an output vector $\mathbf{y}_t$, which

can be used for prediction or other down-stream tasks. Elman-RNN and Jordan-RNN differ in how $\mathbf{h}_t$ are generated. In Elman-RNN, $\mathbf{h}_t$ is calculated as:

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{x}_t + \mathbf{b}). \tag{2.5}$$

where $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{b}$ are parameters to be learned and $f(\cdot)$ is a nonlinear activation function. In Jordan-RNN, $\mathbf{h}_t$ is calculated as:

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{y}_{t-1} + \mathbf{V}\mathbf{x}_t + \mathbf{b}).$$

However, both Elman-RNN and Jordan-RNN suffer from vanishing gradient and exploding gradient problem, which makes them less capable of modeling long-term dependencies. To overcome this probem, Long short-term memory (LSTM) network was proposed [26]. It uses a gate mechanism to control the encoding of sequences. There are 3 gates at each position, namely the forget gate $\mathbf{f}_t$, the input gate $\mathbf{i}_t$ and the output gate $\mathbf{o}_t$, which are generated based on the hidden representation and input data. There is also a cell state $\mathbf{C}_t$, which is used to update the memory state $\mathbf{h}_t$. The full derivation of the gates and states are:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{x}_t + \mathbf{b}_f) && (2.6)\\
\mathbf{i}_t &= \sigma(\mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{x}_t + \mathbf{b}_i) && (2.7)\\
\mathbf{o}_t &= \sigma(\mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{x}_t + \mathbf{b}_f) && (2.8)\\
\hat{\mathbf{C}}_t &= \tanh(\mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{V}_c\mathbf{x}_t + \mathbf{b}_c) && (2.9)\\
\mathbf{C}_t &= \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \hat{\mathbf{C}}_t && (2.10)\\
\mathbf{h}_t &= \mathbf{o}_t * \tanh(\mathbf{C}_t). && (2.11)
\end{aligned}$$

Another type of RNN unit using gate mechanism was also proposed, which is de-

fined as:

$$\mathbf{z}_t = \sigma(\mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{V}_z\mathbf{x}_t + \mathbf{b}_z) \quad (2.12)$$

$$\mathbf{r}_t = \sigma(\mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{V}_r\mathbf{x}_t + \mathbf{b}_r) \quad (2.13)$$

$$\hat{\mathbf{h}}_t = \tanh(\mathbf{W}(\mathbf{r}_t * \mathbf{h}_t) + \mathbf{V}\mathbf{x}_t + \mathbf{b}) \quad (2.14)$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \hat{\mathbf{h}}_t \quad (2.15)$$

It uses less parameters as there are also two gates in it. An empirical comparison demonstrates that both LSTM and GRU can significantly outperform traditional RNN models [13]. But there is no concrete conclusion on which of these two RNN units is better.

Although RNN models have been proved to be much more effective than traditional sequence labelling methods, the trained models are usually very domain specific. In other words, RNN models cannot get decent performances when the testing data comes from a domain that is different from the training data. The main reason is that RNN models still heavily rely on domain specific features. To overcome this problem, we propose two models to incorporate domain independent rules into RNN for aspect word identification in Chapter 4. With the power of supervised RNN model and unsupervised syntactic rules, our model is able to model both explicit and implicit patterns of identifying aspect words. To the best of our knowledge, there has been no neural networks developed to do this.

2.3.3 Neural Networks for Aspect Word Clustering

Since the emergency of LDA [2], various extensions based on it have been proposed. However, most of these models rely on word co-occurrences to learn topics. They share the same assumption that words co-occur frequently in the same documents are more likely to form a topic. Words are treated as discrete signals that are independent from each other in these models. Their semantic meanings are totally ignored. The recent advance in word embeddings [59, 73] and neural networks has

brought us the opportunity to overcome this problem. As word embedding vectors encode the semantic meanings of words, incorporating them into topic models can help us learn more meaningful topics by leveraging both co-occurrence statistics and semantic similarities. Gaussian-LDA [17] is the first work using word vectors to represent words in documents. In LDA, the probability of generating a word given a topic follows a multinomial distribution:

$$P(w|t, \phi_t) = \text{Mult}(\phi_t). \tag{2.16}$$

Gaussian-LDA generates word vectors instead of word indexes. The probability of getting a word vector given a topic follows a multivariate Gaussian distribution:

$$P(\mathbf{v}_w|t, \mu_t, \mathbf{\Sigma}_t) \sim \mathbf{N}(\mu_t, \mathbf{\Sigma}_t). \tag{2.17}$$

Cao et al. [6] designed the Neural Topic Model (NTM) to model documents using a generative process based on neural networks. The most significant difference between NTM and Gaussian-LDA is that NTM models topic word distribution based on word embedding vectors as follows:

$$\phi_{tw} = \frac{\sigma(\mathbf{v}_w^\intercal \mathbf{e}_t)}{\sum_{w'} \sigma(\mathbf{v}_{w'}^\intercal \mathbf{e}_t)} \tag{2.18}$$

where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{e}_t$ is the embedding vector for topic t. Experiments show that both Gaussian-LDA and NTM can outperform traditional LDA in document classification. It proves that incorporating word vectors can help use learn more meaningful topics. Softmax function can also be used to model the word distribution in a topic [66]:

$$\phi_{tw} = \frac{\exp(\mathbf{v}_w^\intercal \mathbf{e}_t)}{\sum_{w'} \exp(\mathbf{v}_{w'}^\intercal \mathbf{e}_t)}. \tag{2.19}$$

This word distribution based on word embedding vectors is also combined with traditional multinomial word distribution in [66]. The hybrid model can outperform traditional topic models in terms of topic coherence and document clustering.

We propose two models based on word embedding vectors to model product review content in Chapter 5 and Chapter 6. Our model in Chapter 5 is a combination of Gaussian-LDA and matrix factorization. We find that modeling product reviews with word vectors can help us do a better job in product recommendation. To the best of our knowledge, no one has tried to use distant supervision to guide the generation of documents based on word embedding vectors. Our model in Chapter 6 tries to tackle this problem by incorporating RNN model for sequence labelling into a neural topic model. RNN model can help us identify words that are most relevant to aspect discovery. Our experiments demonstrate that our proposed hybrid model can outperform existing baselines in terms of perplexity, topic coherence and sentence clustering.

Chapter 3

A Joint Model of Product Properties, Aspects and Ratings for Online Reviews

Before doing aspect discovery, we need to first understand and determine what are considered aspects. However, people do not think too much about this problem and most previous work does not explicitly study this either as they assume that topics learned by topic modeling are latent aspects that people use to evaluate products. In fact, there exist other types of topics besides rateable aspects. Properties are one type of topics that previous work either misses or fails to separate from rateable aspects. In this chapter, we propose a new model on top of Hidden Factors as Topics (HFT) model to separate product properties and aspects. Product properties are intrinsic to certain products (e.g. types of cuisines of restaurants) whereas aspects are dimensions along which products in the same category can be compared (e.g. service quality of restaurants). Our proposed model explicitly separates the two types of latent factors but links both to product ratings. Experiments show that our proposed model is effective in separating product properties from aspects.

3.1 Introduction

Online product reviews and the numerical ratings that come with them have attracted much attention in recent years. During the early years of research on product review mining, there were two separate lines of work. One focused on content analysis using review texts but ignored users, and the other focused on collaborative filtering-based rating prediction using user-item matrices but ignored texts. For content analysis, the objectives of previous studies included predicting the overall sentiment polarity or numerical rating of a review using text only [72, 71], identifying features or aspects of products on which they are evaluated [27, 87], and modeling the feature or aspect level ratings [27, 86, 92]. However, these studies do not consider the identities of reviewers, and thus cannot incorporate user preferences into the models. In contrast, the objective of collaborative filtering-based rating prediction is to predict a target user's overall rating on a target product without referring to any review text [80]. Collaborative filtering makes use of past ratings of the target user, the target item and other user-item ratings to predict the target user's rating on the target item.

Presumably if review texts, numerical ratings, user identities and product identities are analyzed together, we may achieve better results in rating prediction and feature/aspect identification. This is the idea explored in a recent work by [57], where they proposed a model called Hidden Factors as Topics (HFT) to combine collaborative filtering with content analysis. HFT combines latent factor models for recommendation with Latent Dirichlet Allocation (LDA). In the joint model, the latent factors play dual roles: They contribute to the overall ratings, and they control the topic distributions of individual reviews.

While HFT is shown to be effective in both predicting ratings and discovering meaningful latent factors, we observe that the discovered latent factors are oftentimes not "aspects" in which products can be evaluated and compared. In fact, the authors themselves also pointed out that the topics discovered by HFT "are not sim-

ilar to aspects" [57]. Here we use "aspects" to refer to criteria that can be used to compare all or most products in the same category. For example, we can compare restaurants by how well they serve customers, so *service* is an aspect. But we cannot compare restaurants by how well they serve Italian food if they are not all Italian restaurants to begin with, so *Italian food* cannot be considered an aspect. It is more like a feature or property that a restaurant either possesses or does not possess. Identifying aspects would help businesses see where they lose out to their competitors and consumers to directly compare different products under the same criteria. In this chapter, we study how we can modify the HFT model to discover both properties and aspects. We use the term "product properties" or simply "properties" to refer to latent factors that can explain user preferences but are intrinsic to only certain products. Besides types of cuisines, other examples of properties include brands of products, locations of restaurants or hotels, etc. Since a product's rating is related to both the properties it possesses and how well it scores in different aspects, we propose a joint model that separates product properties and aspects but links both of them to the numerical ratings of reviews.

We evaluate our model on three data sets of product reviews. Based on human judgment, we find that our model can well separate product properties and aspects while at the same time maintaining similar rating prediction accuracies as HFT. In summary, the major contribution of our work is a new model that can identify and separate two different kinds of latent factors, namely product properties and aspects.

3.2 Model

In this section, we first briefly review the HFT model. We then describe our joint model for product properties, aspects and ratings. In this section, we will describe our join model for product properties, aspects and ratings.

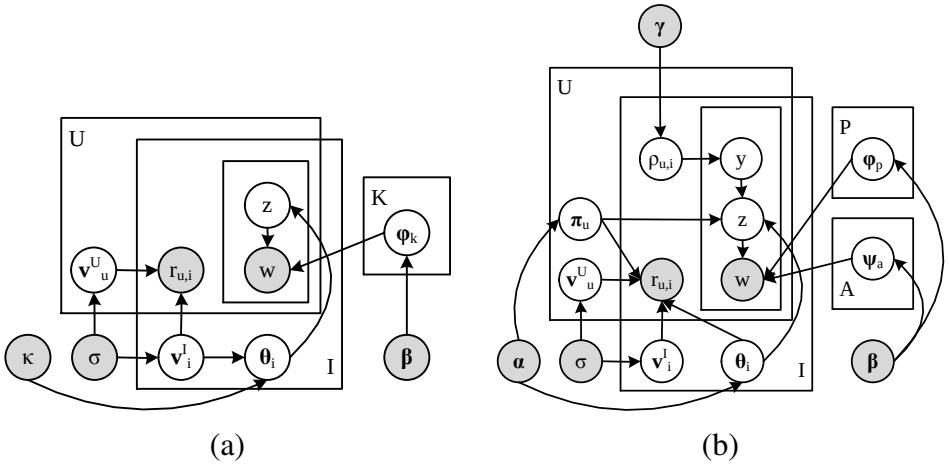


Figure 3.1: Plate notation of (a) HFT and our (b) PAR model. Circles in gray indicate hyperparameters and observations.

3.2.1 The HFT Model

There have been quite a few works on modelling product reviews and ratings simultaneously. However, most of them do not model user preference and item properties like our model. So we picked HFT [57] as our baseline as it could be considered as the state-of-the art method in jointly modelling topics and ratings of review text, user preference and item properties.

As our model can be considered as a extension of HFT model, so we first briefly introduce HFT as background knowledge. The HFT model is one of the state-of-the-art methods that jointly model review texts, ratings, user preferences, and item properties. In this model, it is assumed that there are K latent factors that influence ratings as well as review texts. Items and users each have a real-valued latent vector in the space of the latent factors. The rating $r_{u,i}$ of user u on item i is modeled as the dot product of $\boldsymbol{v}_u^U$ and $\boldsymbol{v}_i^I$ plus some bias terms, where $\boldsymbol{v}_u^U$ is the latent vector for user u and $\boldsymbol{v}_i^I$ the latent vector for item i. This assumption is the same as in previous work on latent factor models for rating prediction.

The major difference of HFT from previous work is that it transforms the item

vector into a probability distribution as follows:

$$\theta_{i,k} = \frac{\exp\left(\kappa v_{i,k}^{I}\right)}{\sum_{k'} \exp\left(\kappa v_{i,k'}^{I}\right)}, \tag{3.1}$$

where $v_{i,k}^{I}$ is the k-th element of latent vector $\boldsymbol{v}_i^I$, $\boldsymbol{\theta_i}$ is the topic distribution of reviews on item i and $\theta_{i,k}$ is the probability of topic k. It then uses the multinomial distribution parameterized by $\boldsymbol{\theta}_i$ to sample latent topics for each word in a review, in the same way as LDA does. Essentially in HFT, each latent topic corresponds to a latent factor. Figure 3.1(a) shows the basic idea of HFT.

It transforms each latent factor to a topic distribution through an exponential transformation. And the corresponding review documents are generated according to this topic distribution. Its objective function is a linear combination of the squared error of rating predictions and log-likelihood of review documents, which can be formally shown as:

$$\mathcal{F}_{\text{HFT}} = \sum_{r_{u,i} \in \mathcal{T}} (rec(u,i) - r_{u,i})^2 - \mu l(\mathcal{T}|\theta, \phi) \tag{3.2}$$

where $rec(u,i)$ is the predicted ratings of item i by user u, $l(\mathcal{T}|\theta,\phi)$ is the log likelihood of review documents and μ is the weighting parameter.

3.2.2 Our Model

Generation of Ratings

As we have pointed out in Section 3.1, many of the latent factors learned by HFT are product properties such as brands, which cannot be used to compare all products in the same category. In order to explicitly model both product properties and aspects, we first assume that there are two different sets of latent factors: There is a set of P product properties, and there is another set of A product aspects. Both are latent factors that will influence ratings.

Next, we assume that each product has a distribution over product properties and

each user has a real-valued vector over product properties. Because properties generally model features that a product either possesses or does not possess, it makes sense to associate a distribution over properties with a product. For example, if each type of cuisines corresponds to a property, then a Mexican restaurant should have a high probability for the property *Mexican food* but low or zero probabilities for properties such as *Japanese food*, *Italian food*, etc. If here we instead use a real-valued vector where negative values are allowed, it would be hard to interpret the vector. For example, for a restaurant which does not serve Mexican food, should it have a zero value or a very negative value for the property *Mexican food*? On the other hand, a user may like and dislike certain product properties, so it makes sense to use real numbers that can be positive or negative to indicate a user's preferences over different properties. For example, if a user does not like Japanese food, she is likely to give low ratings to Japanese restaurants, and therefore it makes sense to model this as a negative value associated with the property *Japanese food* in her latent vector.

Analogically, it makes sense to assume that a product has a real-valued latent vector over aspects, where a positive value means the product is doing well in that aspect and a negative value means the product is poor in that aspect. For example, a restaurant may get a negative score for the aspect *service* but a positive score for the aspect *price*. On the other hand, we assume that a user has a distribution over aspects to indicate their relative weight when the user rates a product. For example, if service is not important to a user but price is, she will have a low or zero probability for the aspect *service* in her vector but a high probability for the aspect *price*. It would not make sense to use a negative value to indicate a user's indifference to an aspect. If a restaurant is doing well in the aspect *service* but service is not important to a user, this should not lower the user's rating to the restaurant.

Formally, let $\boldsymbol{\theta}_i$ denote the property distribution of product i, $\boldsymbol{v}_u^U$ denote the property vector of user u, $\boldsymbol{\pi}_u$ denote the aspect distribution of user u and $\boldsymbol{v}_i^I$ denote

the aspect vector of item i. Based on the assumptions above, it makes sense to model the rating of user u given to item i to be close to $(\boldsymbol{\theta}_i \cdot \boldsymbol{v}_u^U + \boldsymbol{\pi}_u \cdot \boldsymbol{v}_i^I)$. If we compare this formulation with standard ways of modeling ratings such as in HFT, we can see that the major difference is the following. In standard models, the latent vectors of both users and items are unconstrained, i.e. both positive and negative values can be taken. This may cause problem interpreting the learned vectors. For example, when user u has a negative value for the k^{th} latent factor and item i also has a negative value for the k^{th} latent factor, the product of these two negative values results in a positive contribution to the rating of item i given by user u. But how shall we interpret these two negative values and their combined positive impact to the rating? In our model, we separate the latent factors into two groups. For one group of latent factors (product properties), we force the items to have non-negative values, while for the other group of latent factors (product aspects), we force the users to have non-negative values. By doing this, we improve the interpretability of the learned latent vectors.

Generation of Review Texts

In our model, for each latent factor, which can be either a product property or an aspect, there is a word distribution associated with it, which we denote by $\boldsymbol{\phi}_p$ for property p and $\boldsymbol{\psi}_a$ for aspect a.

We assume that a review of a product given by a particular user mainly consists of two types of information: properties this product possesses and evaluation of this product in the various aspects that this user cares about. Content related to product properties is mainly controlled by the property distribution of the product. For example, reviews on a Mexican restaurant may contain much information about Mexican food. Content related to aspects are mainly controlled by the user's aspect preference distribution. A user who values service more may comment more about a restaurant's service. Based on these assumptions, in the generative process of reviews, each word in a review document is sampled either from a product property

or an aspect.

The Generative Process

Our model is shown in Figure 3.1. and the description of the generative process is as follows:

- For each product property p, sample a word distribution $\boldsymbol{\phi}_p \sim$ Dirichlet($\boldsymbol{\beta}$).
- For each aspect a, sample a word distribution $\boldsymbol{\psi}_a \sim$ Dirichlet($\boldsymbol{\beta}$).
- For each item
  - Sample a product property distribution $\boldsymbol{\theta}_i \sim$ Dirichlet($\boldsymbol{\alpha}$).
  - Sample an A-dimensional vector $\boldsymbol{v}_i^I$ where $v_{i,a}^I \sim$ Normal$(0, \sigma^2)$.
  - Sample an item rating bias $b_i \sim \mathcal{N}(0, \sigma^2)$.
- For each user
  - Sample an aspect distribution $\boldsymbol{\pi}_u \sim$ Dirichlet($\boldsymbol{\alpha}$).
  - Sample a P-dimensional vector $\boldsymbol{v}_u^U$ where $v_{u,p}^U \sim$ Normal$(0, \sigma^2)$.
  - Sample a user rating bias $b_u \sim \mathcal{N}(0, \sigma^2)$.
- For a user-item pair where a review and a rating exist
  - Sample the rating $r_{u,i} \sim$ Normal$(\boldsymbol{\theta}_i \cdot \boldsymbol{v}_u^U + \boldsymbol{\pi}_u \cdot \boldsymbol{v}_i^I + b_i + b_u, \sigma^2)$
  - Sample the parameter for a Bernoulli distribution $\boldsymbol{\rho}_{u,i} \sim$ Beta($\boldsymbol{\gamma}$)
  - For each word in the review
    - Sample $y_{u,i,n} \sim$ Bernoulli($\boldsymbol{\rho}_{u,i}$).
    - Sample $z_{u,i,n} \sim$ Discrete($\boldsymbol{\theta}_i$) if $y_{u,i,n} = 0$ and $z_{u,i,n} \sim$ Discrete($\boldsymbol{\pi}_u$) if $y_{u,i,n} = 1$.
    - Sample $w_{u,i,n} \sim$ Discrete($\boldsymbol{\phi}_{z_{u,i,n}}$) if $y_{u,i,n} = 0$ and $w_{u,i,n} \sim$ Discrete($\boldsymbol{\psi}_{z_{u,i,n}}$) if $y_{u,i,n} = 1$.

Here, $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are hyper-parameters for Dirichlet distribution, σ is the standard deviation for Gaussian distribution, $\boldsymbol{\rho}_{u,i}$ is the switching probability distribution for

review of user u on item i, $y_{u,i,n}$ and $z_{u,i,n}$ are the switching variable and topic assignment for word at position n of review on itme i from user u. We refer to our model as the Property-Aspect-Rating (PAR) model.

3.2.3 Parameter Estimation

Our goal is to learn the parameters that can maximize the log-likelihood of both review texts and ratings simultaneously. Formally speaking, we are trying to estimate the parameters $\boldsymbol{V}^U$, $\boldsymbol{V}^I$, $\boldsymbol{B}_U$, $\boldsymbol{B}_I$, $\boldsymbol{\pi}_U$, $\boldsymbol{\theta}_I$, $\boldsymbol{\rho}$, $\boldsymbol{\phi}_P$ and $\boldsymbol{\psi}_A$ that can optimize the following posterior probability.

$$
\begin{aligned}
& P(\boldsymbol{V}^U, \boldsymbol{V}^I, \boldsymbol{B}_U, \boldsymbol{B}_I, \boldsymbol{\pi}_U, \boldsymbol{\theta}_I, \boldsymbol{\rho}, \boldsymbol{\phi}_P, \boldsymbol{\psi}_A | \boldsymbol{W}, \boldsymbol{R}) \qquad (3.3) \\
= \; & \frac{P(\boldsymbol{V}^U, \boldsymbol{V}^I, \boldsymbol{B}_U, \boldsymbol{B}_I, \boldsymbol{\pi}_U, \boldsymbol{\theta}_I, \boldsymbol{\rho}, \boldsymbol{\phi}_P, \boldsymbol{\psi}_A, \boldsymbol{W}, \boldsymbol{R})}{P(\boldsymbol{W}, \boldsymbol{R})} \\
\propto \; & P(\boldsymbol{V}^U, \boldsymbol{V}^I, \boldsymbol{B}_U, \boldsymbol{B}_I, \boldsymbol{\pi}_U, \boldsymbol{\theta}_I, \boldsymbol{\rho}, \boldsymbol{\phi}_P, \boldsymbol{\psi}_A, \boldsymbol{W}, \boldsymbol{R}) \\
= \; & P(\boldsymbol{V}^U) P(\boldsymbol{V}^I) P(\boldsymbol{B}^U) P(\boldsymbol{B}^I) P(\boldsymbol{\pi}_U) P(\boldsymbol{\theta}_I) P(\boldsymbol{\rho}) P(\boldsymbol{\phi}_P) P(\boldsymbol{\psi}_A) \\
& \cdot P(\boldsymbol{W} | \boldsymbol{\pi}_U, \boldsymbol{\theta}_I \boldsymbol{\rho}) P(\boldsymbol{R} | \boldsymbol{V}^U, \boldsymbol{V}^I, \boldsymbol{B}^U, \boldsymbol{B}^I)
\end{aligned}
$$

where

$$P(\boldsymbol{v}_u^U) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}), P(\boldsymbol{v}_i^I) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}) \qquad (3.4)$$

$$P(b_u) = \mathcal{N}(0, \sigma_b), P(b_i) = \mathcal{N}(0, \sigma_b) \qquad (3.5)$$

$$P(\boldsymbol{\pi}_u) = \text{Dir}(\boldsymbol{\alpha}), P(\boldsymbol{\theta}_i) = \text{Dir}(\boldsymbol{\alpha}) \qquad (3.6)$$

$$P(\boldsymbol{\rho}_{u,i}) = \text{Beta}(\boldsymbol{\gamma}), P(\boldsymbol{\phi}_p) = \text{Dir}(\boldsymbol{\beta}) \qquad (3.7)$$

$$P(\boldsymbol{\psi}_a) = \text{Dir}(\boldsymbol{\beta}) \qquad (3.8)$$

and

$$P(\boldsymbol{W}|\boldsymbol{\pi}_U, \boldsymbol{\theta}_I, \boldsymbol{\rho}) = \prod_u \prod_i \prod_{w \in d_{ui}} (\boldsymbol{\rho}^0_{u,i} \sum_t \boldsymbol{\theta}_{it} \boldsymbol{\phi}_{tw} + \boldsymbol{\rho}^1_{u,i} \sum_t \boldsymbol{\pi}_t \boldsymbol{\psi}_{tw}) \tag{3.9}$$

$$P(\boldsymbol{R}|\boldsymbol{V}^U, \boldsymbol{V}^I, \boldsymbol{B}^U, \boldsymbol{B}^I) = \prod_{\{u,i\} \in \boldsymbol{D}} \mathcal{N}(r_{ui}|\mathbf{v}_i^{I\intercal} \mathbf{v}_u^U + b_u + b_i, \sigma) \tag{3.10}$$

where Dir$(\cdot)$ denotes a Dirichlet distribution and $\mathcal{N}(\cdot)$ denotes a Gaussian distribution. Besides, $\boldsymbol{V}^U$ and $\boldsymbol{V}^I$ refer to all latent vectors for items and users, $\boldsymbol{B}_U$ and $\boldsymbol{B}_I$ refer to all the bias terms, $\boldsymbol{W}$ refers to all the words in the reviews and $\boldsymbol{R}$ refers to all the ratings. The hyperparameters are omitted in the formula. Equivalently, we will use the loglikelihood as our objective function.

As there is no closed form solution for it, we use Gibbs-EM algorithm [89] for parameter estimation.

E-step: In the E-step, we fix the parameters $\boldsymbol{\pi}_U$ and $\boldsymbol{\theta}_I$ and collect samples of the hidden variables $\boldsymbol{Y}$ and $\boldsymbol{Z}$ to approximate the distribution $P(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{W}, \boldsymbol{R}, \boldsymbol{\pi}_U, \boldsymbol{\theta}_I)$. For each word, we use Gibbs sampling algorithm to sample its latent variable y and z. For word w in a review from user u on item i, the sampling probabilities are:

$$P(y = 0, z = t|\boldsymbol{Y}_{-y}, \boldsymbol{Z}_{-z}, \boldsymbol{W}) \propto \boldsymbol{\rho}^0_{u,i} \boldsymbol{\theta}_{it} \boldsymbol{\phi}_{tw} \tag{3.11}$$

$$P(y = 1, z = t|\boldsymbol{Y}_{-y}, \boldsymbol{Z}_{-z}, \boldsymbol{W}) \propto \boldsymbol{\rho}^1_{u,i} \boldsymbol{\pi}_{ut} \boldsymbol{\psi}_{tw} \tag{3.12}$$

M-step: In the M-step, with the collected samples of $\boldsymbol{Y}$ and $\boldsymbol{Z}$, we seek values of $\boldsymbol{\pi}_U$, $\boldsymbol{\theta}_I$, $\boldsymbol{V}^U$, $\boldsymbol{V}^I$, $\boldsymbol{B}_U$ and $\boldsymbol{B}_I$ that maximize the following objective function:

$$\mathcal{L} = \sum_{(\boldsymbol{Y}, \boldsymbol{Z}) \in \mathcal{S}} \log P(\boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{R}|\boldsymbol{\pi}_U, \boldsymbol{\theta}_I, \boldsymbol{V}^U, \boldsymbol{V}^I, \boldsymbol{B}_U, \boldsymbol{B}_I) \tag{3.13}$$

where $\mathcal{S}$ is the set of samples collected in the E-step. The gradient for these param-

eters are calculated as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_u^U} = \sum_{\{u,i\}\in\mathcal{D}} e_{ui}\boldsymbol{\theta}_i \quad , \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_i^I} = \sum_{\{u,i\}\in\mathcal{D}} e_{ui}\boldsymbol{\pi}_u \tag{3.14}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_u} = \sum_{\{u,i\}\in\mathcal{D}} e_{ui} \quad , \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_i} = \sum_{\{u,i\}\in\mathcal{D}} e_{ui}. \tag{3.15}$$

where $\mathcal{D}$ is the union of all user-item pairs with ratings and reviews in our dataset.

As $\boldsymbol{\theta}_i$ and $\boldsymbol{\pi}_u$ are two stochastic vectors, we cannot directly use gradient descent based algorithm to find their optimal values. We replace these parameters with two exponential transformations of some auxiliary parameters by:

$$\boldsymbol{\theta}_{it} = \frac{\exp(\eta_{it})}{\sum_t \exp(\eta_{it})} \quad , \quad \boldsymbol{\pi}_{ut} = \frac{\exp(\kappa_{ut})}{\sum_t \exp(\kappa_{ut})}. \tag{3.16}$$

The derivatives with respect to these auxiliary parameters are:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \eta_{it}} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_{it}} \frac{\partial \boldsymbol{\theta}_{it}}{\partial \eta_{it}} \\ &= (N_t^I + \alpha)(1 - \boldsymbol{\theta}_{it}) + \frac{2}{\sigma} \sum_{\{u,i\}\in\mathcal{D}} e_{ui}\boldsymbol{v}_{ut}^U \boldsymbol{\theta}_{it}(1 - \boldsymbol{\theta}_{it}) \end{aligned} \tag{3.17}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \kappa_{ut}} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{\pi}_{ut}} \frac{\partial \boldsymbol{\pi}_{ut}}{\partial \kappa_{ut}} \\ &= (N_t^U + \alpha)(1 - \boldsymbol{\pi}_{ut}) + \frac{2}{\sigma} \sum_{\{u,i\}\in\mathcal{D}} e_{ui}\boldsymbol{v}_{it}^I \boldsymbol{\pi}_{ut}(1 - \boldsymbol{\pi}_{ut}) \end{aligned} \tag{3.18}$$

where e_{ui} is the error of predicting rating $r_u i$, which can be calculated as

$$\boldsymbol{\theta}_i^\intercal \boldsymbol{v}_u^U + \boldsymbol{\pi}_u^\intercal \boldsymbol{v}_i^I + b_i + b_u - r_{ui}. \tag{3.19}$$

In our implementation, we perform 600 runs of Gibbs EM. Because Gibbs sampling is time consuming, in each run we only perform one iteration of Gibbs sampling and collect that one sample. We then have 60 iterations of gradient descent. The gradient descent algorithm we use is L-BFBS, which is efficient for large scale data set.

Data Set	#Reviews	#W/R	Voc	#Users	#Items
SOFT	54,330	84.6	16,653	43,177	8,760
MP3	20,689	103.9	8,227	18,609	742
REST	88,865	86.5	21,320	8,230	3,395

Table 3.1: Statistics of our data sets.*#W/R stands for #Word/Review and Voc stands for vocabulary.

3.3 Experiments

In this section, we present the empirical evaluation of our model.

3.3.1 Data

We use three different review data sets for our evaluation. The first one is a set of software reviews, which was used by [57]. We refer to this set as SOFT. The second one is a set of reviews of MP3 players, which was used by [92]. We refer to this set as MP3. The last one is a set of restaurant reviews released by Yelp[1] in Recsys Challenge 2013[2], which was also used by [57]. We refer to it as REST. Based on common practice in previous studies [86, 87, 90], we processed these reviews by first removing all stop words and then removing words which appeared in fewer than 10 reviews. We then also removed reviews with fewer than 30 words. Some statistics of the processed data sets are shown in Table 3.1.

3.3.2 Experiment Setup

As we have discussed in Section 3.1, the focus of our study is to modify the HFT model to capture both product properties and aspects. Note that HFT model is designed for both predicting ratings and discovering meaningful latent factors. Therefore, the goal of our evaluation is to test whether our PAR model can perform similarly to HFT in terms of rating prediction and latent factor discovery, and on top of that, whether our PAR model can well separate product properties and aspects,

[1]http://www.yelp.com
[2]https://www.kaggle.com/c/yelp-recsys-2013

which HFT cannot do. In the rest of this section, we present our evaluation as follows. We first compare PAR with HFT in terms of finding meaningful latent factors. We then evaluate how well PAR separates properties and aspects. Finally, we compare PAR with HFT for rating prediction. Note that when we compare PAR with HFT in the first and the third tasks, we do not expect PAR to outperform HFT but we want to make sure PAR performs comparably to HFT.

In all our experiments, we use the same number of latent factors for PAR and HFT. For PAR, the number of latent factors is the number of properties plus the number of aspects, i.e. $P + A$. After some preliminary experiments, we set the total number of latent factors to 30 for both models. For PAR, based on observations with the preliminary experiments, we empirically set P to 10 and A to 20. User latent factor $\boldsymbol{v}_u^U$ and item latent factor $\boldsymbol{v}_i^I$ should have the dimensions equal to P and A respectively. Although these settings may not be optimal, by using the same number of latent factors for both models, no bias is introduced into the comparison.

For other hyperparameters, we empirically tune the parameters using a development set and use the optimal settings. For PAR, we set $\alpha = 2$, $\beta = 0.01$, $\sigma = 0.1$ and $\gamma = 1$. For HFT, we set $\mu = 10$ for MP3 and SOFT and $\mu = 0.1$ for REST. All results reported below are done under these settings.

3.3.3 Annotation of Ground Truth

The major goal of our evaluation is to see how well the PAR model can identify and separate product properties and aspects. However, in all three data sets we use, there is no ground truth and we are not aware of any data set with ground truth labels we can use for our task. Therefore, we have to annotate the data ourselves.

Instead of asking annotators to come up with product properties and aspects, which would require them to manually go through all reviews and summarize them, we opted to ask them to start from latent factors discovered by the two models. We randomly mixed the latent factors learned by PAR and HFT. The top 15 words of

	Product Properties		Aspects	
	Number	Avg. # Relevant Words	Count	Avg. # Relevant Words
SOFT	18	11.3	9	9.2
MP3	6	5.0	13	9.9
REST	13	10.4	5	7.8

Table 3.2: Summary of the ground truth latent factors.

	SOFT			MP3			REST		
	# Good LF	Prec	Rec	# Good LF	Prec	Rec	# Good LF	Prec	Rec
PAR	20	0.67	0.74	14	0.47	0.74	10	0.33	0.56
HFT	20	0.67	0.74	12	0.40	0.63	10	0.33	0.56

Table 3.3: Results for identification of meaningful latent factors

each latent factor were shown to two annotators, and each annotator independently performed the following three steps of annotations. In the first step, an annotator had to determine whether a latent factor was meaningful or not based on the 15 words. In the second step, for latent factors labeled as meaningful, an annotator had to decide whether it was a product property or an aspect. In the third step, an annotator had to pick relevant words from the given list of 15 words for each latent factor. After the three-step independent annotation, the two annotators compared and discussed their results to come to a consensus. During this discussion, duplicate latent factors were merged and word lists for each latent factor were finalized. The annotators were required to exclude general words such that no two latent factors share a common relevant word. In the end, the annotators produced a set of product properties and another set of aspects for each data set. For each latent factor, a list of highly relevant words was also produced. Table 3.2 shows the numbers of ground truth properties and aspects as labeled by the annotators and the average numbers of relevant words per latent factor of the three data sets.

3.3.4 Discovery of Meaningful Latent Factors

In the first set of experiments, we would like to compare PAR and HFT in terms of how well they can discover meaningful latent factors. Here latent factors include both product properties and aspects.

Results

We show three numbers for each data set and each method. The first is the number of "good" latent factors discovered by a method. Here a good latent factor is one that matches one of the ground truth latent factors. A learned latent factor matches a ground truth latent factor if the top-15 words of the learned latent factor cover at least 60% of the ground truth relevant words of the ground truth latent factor. We find the 60% threshold reasonable because most matching latent factors appear to be meaningful.

We use Precision and Recall as the evaluation metric. We would like to point out that the recall defined in this way is higher than the real recall value, because our ground truth latent factors all come from the discovered latent factors, but there may exist meaningful factors that are not discovered by either HFT or PAR at all. Nevertheless, we can still use this recall to compare PAR with HFT. The results are shown in Table 3.3. As we can see from the table, PAR and HFT performed similarly in terms of discovering meaningful latent factors. PAR performed slightly better than HFT on the MP3 data set. Overall, between one-third to two-thirds of the discovered latent factors are meaningful for both methods, and both methods can discover more than half of the ground truth latent factors.

3.3.5 Separation of Product Properties and Aspects

In this second set of experiments, we would like to evaluate how well PAR can separate product properties and aspects. In order to focus on this goal, we first disregard the discovered latent topics that are not considered good latent topics according to the criterion used in the previous experiment.

We then show the 2×2 confusion matrix between the labeled two types of latent factors and the predicted two types of latent factors by PAR for each data set. The results are in Table 3.4. As we can see, our model does a very good job in separating the two types of latent factors for MP3 and REST. For SOFT, our model mistakenly

Prediction	Ground Truth					
	SOFT		MP3		REST	
	P	A	P	A	P	A
P	8	2	3	0	8	0
A	4	6	1	10	0	2

Table 3.4: Confusion matrices of PAR for all data sets. *P stands for property and A stands for aspect.

labeled 4 product properties as aspects. Although this result is not perfect, it still shows that our model can separate properties from aspects well in different domains.

We find that properties in the software domain are mostly functions and types of software such as games, antivirus software and so on. Aspects of software include software version, user interface, online service and others. In the MP3 data set, properties are mainly about MP3 brands such as Sony and iPod while aspects are about batteries, connections with computers and some others. Properties of the restaurant data set are all types of cuisines and aspects include ambiance and service.

In order to qualitatively understand the performance of our model on separating aspects from product properties, we show the aspects and properties of SOFT learned by our model in Table 3.5. We can see that aspects are mainly topics over which users can evaluate a product, like user experience, interface, customer service and so on. Properties are about the type of software and their brands. The observation follows our hypothesis and proves that our model is effective in separating aspects from product properties.

3.3.6 Rating Prediction

Finally we compare our model with HFT for rating prediction in terms of root mean squared error. The results are shown in Table 3.6. We can see that PAR outperforms HFT in two real data sets (SOFT, MP3) and gets the same performance for the data set REST. This means separating properties and aspects in the model did not compromise rating prediction performance, which is important because otherwise the learned latent factors might not be the best ones explaining the ratings.

Properties	File Backup	drive backup hard disk linux files system data partition suse boot ghost cd computer image
	Microsoft Office	office word microsoft ms use excel outlook documents document pdf
	Games	game games play fun old kids get one playing son year child like daughter played
	Tax Preparation	tax year turbotax state taxcut return years taxes software turbo
	CD/DVD Burning	cd dvd software roxio creator burn easy nero copy disc
	Windows OS	windows xp vista system computer microsoft install run os upgrade new
	Financial	quicken money data business quickbooks intuit account accounts microsoft years
	Antivirus	norton computer internet system security mcafee virus firewall symantec antivirus
Aspects	Version	version new years since old upgrade still versions used previous
	Experience	screen feature file click change list open option see set
	User Interface	user interface features users available application system many ability offers
	Purchase	software product buy price purchased reviews purchase amazon recommend products
	Online Service	web free site support page website online help get read
	Edition	simple home design tool create make basic want professional limited

Table 3.5: The properties and aspects of SOFT learned by PAR. The second column is the human annotated labels of each latent factor and the third column is the top 10 words of the corresponding latent factor. We can see that most properties are types of software.

	SOFT	REST	MP3
PAR	1.394	1.032	1.401
HFT	1.399	1.032	1.404

Table 3.6: Performance in rating prediction.

3.3.7 Latent Factors Learned by PAR

To compare the performance of identifying meaningful latent factors, we pick out the latent factors learned by PAR and HFT that are easy to interpret and show them in Table 3.7. The SOFT and MP3 data sets are used and each latent factor is manually labeled by the authors. We can see that PAR can discover more meaningful aspects than HFT in both datasets. However, HFT learns some aspects that are not covered by PAR, like network, which needs to be addressed in our following up work.

3.4 Discussion

We presented a joint model of product properties, aspects and numerical ratings for online product reviews. The major advantage of the proposed model is its ability to separate product properties, which are intrinsic to products, from aspects that are meant for comparing products in the same category. To achieve this goal, we combined probabilistic topic models with matrix factorization. We explicitly separated the latent factors into two groups and used both groups to generate both review texts and ratings. Our evaluation showed that compared with HFT our model could achieve similar or slightly better performance in terms of identifying meaningful latent factors and predicting ratings. More importantly, our model is able to separate product properties from aspects, which HFT and other existing models are not capable of.

SOFT		
PAR	Operating System	mac os apple pc act use windows work run leopard running application virtual time using
	Upgrade	version new years since old upgrade still versions used previous ago features using much year
	User Interface	screen feature file click change list open option see set start window options name menu
	Experience	user interface features users available application system many ability offers using friendly server provides functions
	Purchase	software product buy price purchased reviews purchase amazon recommend products package buying company disappointed cost
	Online Service	web free site support page website online help get read download box number pay manual
	Support	support software product problem work tech get would problems tried customer install computer installed phone
	Version	simple home design tool create make basic want professional limited plan allows tools also etc
HFT	Operating System	act server database outlook data using access palm network email business microsoft manager features sync
	Network	mac os apple pc windows use leopard new run work imac tiger parallels upgrade virtual

MP3		
PAR	Problem	software computer problem work get manual problems tried would player first download time firmware could
	Connection with computer	usb windows device pc software computer media use 2. 1. files xp cable 3. sync
	Controls	volume buttons button menu screen use sound good control like wheel touch scroll back easy
	Battery	battery hours life charge usb charger unit use power long 2 time charging 3 adapter
	Music	album songs song artist playlist playlists albums tracks play name music list order art track
	Audio	fm radio recording record voice tuner features recorder also sound good feature unit quality reception
	Video	video videos screen movies tv watch music play great pictures quality movie photos picture convert
	Sound Quality	sound player quality great mp3 good headphones use easy sony ear better product excellent recommend
	Size	nano ipod screen case mini small black scratches color easily like scratch size great pocket
	Service	service customer support amazon product unit back would get one days called told new send
	Photo	photo photos drive palm pictures hard device lifedrive screen camera pda color view life storage
HFT	Memory	card sd memory cards unit slot firmware speaker flash expansion display file back internal files
	Audio	fm radio player mp3 recording voice record unit recorder tuner good features also sound well
	Network	touch classic wifi screen internet iphone web device wireless browser itouch bluetooth use wi-fi pad
	Video	video screen videos movies files watch tv photos picture pictures media movie play convert format
	Sound Quality	sound quality great good use like better headphones battery life best audio listening small really
	Size	nano small screen 4gb black color 8gb scratches 2gb much memory case size pocket 4

Table 3.7: Ratable aspect topics and their top words identified by PAR and HFT in SOFT and MP3 datasets. The first column shows the methods while the second column and third column is the topic and topic word list respectively.

Chapter 4

Recurrent Neural Networks with Auxiliary Labels for Cross-domain Opinion Target Extraction

As introduced in Chapter 1, the second step of aspect discovery from product reviews is to identify words that are describing aspects. Since we mainly care about aspects users are showing opinions to in product reviews, aspect word identification can also be treated as an opinion target extraction problem. In recent years, neural network based supervised learning methods have achieved competitive performance on this task. However, as with any supervised learning method, neural network based methods for this task cannot work well when the training data comes from a different domain than the test data. On the other hand, some rule-based unsupervised methods haven shown to be robust when applied to different domains. In this chapter, we use rule-based unsupervised methods to create auxiliary labels and use neural network models to learn a hidden representation that works well for different domains. When this hidden representation is used for opinion target extraction, we find that it can outperform a number of strong baselines with a large margin.

4.1 Introduction

Opinion Target extraction is one of the most fundamental problems in opinion mining. It is important to many down stream applications, such as sentiment prediction, opinion summarization and so on. This problem has attracted enormous attention from research community in the last decade [83]. Given a sentence, its goal is to extract the targets that users are expressing opinion towards. For example, given the sentence *I like the tuna sandwich and chicken salad very much.*, the correctly extracted targets should be *tuna sandwich* and *chicken salad*. This is typically modeled as a sequence labeling problem and Conditional Random Field(CRF) [43] is one of the most popular techniques used for it. CRF is an undirected probabilistic graphical model that can take various features as input. However, it relies on manual created features that take a lot of time and effort to extract. The recent advances of deep neural networks are bringing new improvement over existing models. Besides, deep neural networks are also able to automatically construct features from the raw input. They are so attractive that many researcher are trying to apply them on multiple NLP problems. Among these deep learning models, Recurrent Neural Network (RNN) and its variations, such BiRNN, LSTM and so on, have been proved to be effective [51, 101, 98, 93] for target extraction.

However, it requires a large amount of labeled data to train neural networks. Creating labeled data can be tedious and time consuming. Opinion target extraction is a very domain sensitive problem. Those neural networks that are well trained based on domain A may not be useful at all in a different domain B. One of the most serious problems causing this is that in different domains, the target words and opinion words may be very different. For example, *food* and *beverage* are frequent opinion targets in restaurant domain while *delicious* and *tasty* are usually the opinion words used on these words. However, in laptop domain, you may not see these words at all. Istead, target words like *CPU* and *hard disk* and opinion expression *fast*, *easy to use* are important and frequent. So a neural network trained on domain

A can only learn lexical and syntactic information that are useful in the itself, which may not generatlize to domain B. People usually create a new labeled dataset when they need to work on a new domain. This may help us build domain specifically effective neural networks, but it also costs a lot of labours. It would be nice if we can train a neural network that can work on different domains efficiently. This is named as "cross-domain target extraction", which has been studied in several previous work. However, most of them are using traditional sequence labeling technique (e.g. CRF).

There exist some natural relations between opinion expressions and opinion targets [77]. Many of them are used in unsupervised target extractions [27, 77]. However, both precision and recall of these techniques can be limited. Parsing tree can be inaccurate, especially for user generated content, which is normally written in an informal way. Secondly, some opinion are expressed explicitly, without using opinion words or common patterns. The targets of these opinions will not be detected by patterns of relations.

It is presumably useful to combine advanced supervised sequence labeler – RNN – with unsupervised domain independent syntactic rules for cross domain target extraction. While syntactic rules can help use learn domain independent representations, labeled training data can tune RNN to be able to model explicit patterns. A combination of these two types of knowledge can help us learn a domain independent representation that will lead to better performance. In this chapter, we design two models to incorporate rules into supervised neural network. We first use domain general rules to assign auxiliary labels to our data. Our neural networks can be trained based on the true labels of source domain data and the auxiliary labels. Experiments over four datasets from different domains show that our methods can outperform the baselines with a large margin.

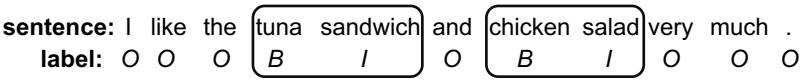


Figure 4.1: A sample sentence and its labels.

4.2 Methodology

4.2.1 Notation

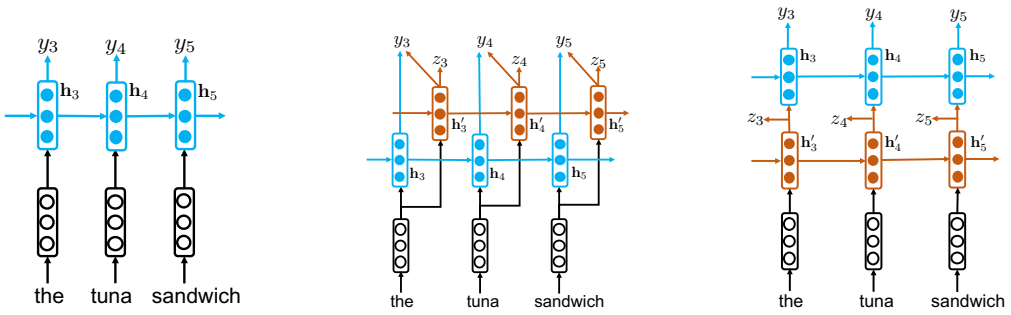
Opinion target extraction aims at extracting all the opinion targets from a given sentence. An opinion target is not restricted to a single token; in fact, it often contains multiple tokens. The task is therefore a typical sequence labeling problem.

Formally, we represent a sentence as a sequence of tokens $\mathbf{x} = (w_1, w_2, \ldots, w_N)$, where each w_i is a word type from a vocabulary $\mathcal{V}$. Opinion targets are indicated by token-level labels $\mathbf{y} = (y_1, y_2, \ldots, y_N)$, where each $y_i \in \{B, I, O\}$. The three labels B, I and O refer to the beginning, inside and outside of an opinion target, respectively, and they follow the standard BIO notation used in sequence labeling. A sample review sentence together with its opinion target labels are shown in Figure 4.1.

We assume that there is a set of labeled review sentences from a *source* domain, denoted with $\mathcal{D}^s = \{(\mathbf{x}^s, \mathbf{y}^s)\}$. On the other hand, the sentences from which opinion targets need to be extracted come from a different *target* domain and are denoted with $\mathcal{D}^t = \{\mathbf{x}^t\}$. We would like to use both $\mathcal{D}^s$ and $\mathcal{D}^t$ to train a good model for opinion target extractions for the target domain.

4.2.2 Overview of Our Method

Our method is essentially a supervised method based on recurrent neural networks. The key to our method is a hidden layer of the neural network that is trained using auxiliary labels created by domain-independent rules. The idea of using auxiliary labels to induce a representation for domain adaptation is not new [4, 3]. It essentially follows the principle of multi-task learning, where it is generally believed



(a) Standard (b) Ours (Concatenated Network) (c) Ours (Hierarchical Network)

Figure 4.2: Overview of the standard method and our method.

that if multiple prediction tasks are related, then the underlying prediction models are likely to share some common feature structures. When the auxiliary tasks are related to the actual prediction task and the labels of the auxiliary tasks can be easily obtained for both the source and the target domains, we can use the auxiliary tasks to help us induce a good hidden feature representation that is good for domain adaptation.

Neural network models are intrinsically suitable for this kind of multi-task learning based domain adaptation because we can naturally use one of the hidden layers as the cross domain hidden representation. To the best of our knowledge, however, there has not been any work on extending neural network models to solve the domain adaptation problem for opinion target extraction. In our method, we use a recurrent neural network (specifically, an LSTM) to process an input sentence such that each token has a corresponding hidden vector. Typically this hidden vector will then go through a linear transformation followed by a softmax layer to make the final prediction. In our method, this hidden vector is used for predicting not only the opinion target label but also some auxiliary label. The auxiliary labels are predicted by manually-crafted syntactic rules, which will be detailed below.

Figure 4.2 illustrates the main idea of our method at a high level. We can see that typically, as shown in Figure 4.2(a), the hidden layer $\mathbf{h}$ is learned only through back propagation from the true labels $\mathbf{y}$, which are only available in the source domain.

With our method, as shown in Figure 4.2(b) and Figure 4.2(c), there is an auxiliary hidden layer $\mathbf{h}'$ that is learned using some auxiliary labels $\mathbf{z}$. And this auxiliary hidden layer $\mathbf{h}'$ is then either to be concatenated with $\mathbf{h}$ or to generate $\mathbf{h}$ in order to predict the true labels $\mathbf{y}$. Because the auxiliary labels $\mathbf{z}$ are available in both the source and the target domains, we can expect the auxiliary hidden layer $\mathbf{h}'$ to be properly learned such that it works well for both domains.

4.2.3 Recurrent Neural Networks for Opinion Target Extraction

In this section, we describe how we use recurrent neural networks for opinion target extraction. Note that this is a standard approach and has been studied before [51]. Recall that we use $\mathbf{x} = (w_1, w_2, \ldots, w_N)$ to represent an input sentence. Let $\mathbf{x}_i \in \mathbb{R}^d$ denote the embedding vector for word w_i. A recurrent neural network model aims at learning a hidden vector representation for each position i that supposedly encodes all the tokens from the beginning of the sentence up to position i. Specifically, let $\mathbf{h}_{i-1}$ denote such a hidden vector corresponding to position $i-1$. Then the hidden vector $\mathbf{h}_i$ is defined as follows:

$$\mathbf{h}_i = g(\mathbf{h}_{i-1}, \mathbf{x}_i), \tag{4.1}$$

where $g(\cdot)$ is some function. For example, in standard RNN, we have

$$\mathbf{h}_i = f(\mathbf{U}\mathbf{h}_{i-1} + \mathbf{V}\mathbf{x}_i + \mathbf{c}), \tag{4.2}$$

where $\mathbf{U} \in \mathbb{R}^{l \times l}$ and $\mathbf{V} \in \mathbb{R}^{l \times d}$ are weight matrices, $\mathbf{c} \in \mathbb{R}^l$ is a bias vector, l is the dimension of the hidden layer, and $f(\cdot)$ is an element-wise non-linear transformation function.

In our experiments, we experiment with a few different types of RNNs, including the standard RNN, bi-directional RNN, long short-term memory (LSTM) network (which is a special form of RNN), and bi-directional LSTM. We will not give

the details of LSTM here. Interested readers can refer to [26] for details.

To simplify the discussion, we use Θ to denote all the parameters used in any type of an RNN, and represent the hidden layer as

$$\mathbf{h}_i = \mathit{RNN}_{\Theta}(\mathbf{h}_{i-1}, \mathbf{x}_i). \tag{4.3}$$

The vector $\mathbf{h}_i$ is then used to predict the opinion target label as follows:

$$p(y_i \mid \mathbf{h}_i) = \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{b}), \tag{4.4}$$

where $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias vector, both to be learned.

4.2.4 Rule-based Auxiliary Labels

Our preliminary experiments using the RNN model presented above on target opinion extraction suggest that the supervised RNN model relies much on lexical information. This is not surprising because the RNN model does not model syntactic structures of a sentence such as part-of-speech tags and dependency relations. Although lexical information is very important for opinion target extraction, it is also very domain specific. As a result, we find that the RNN model performs poorly in cross-domain settings.

On the other hand, people have studied how to use general syntactic patterns to detect opinion targets [105, 77]. An important observation is that opinion targets often co-occur with explicit opinion expressions, which usually contain opinion words. Syntactically, there are some patterns between opinion words and opinion targets, and these patterns tend to be general across different domains. For example, usually the object of the verb *love* is an opinion target. Using this rule, we can predict that the phrases *tuna sandwich* and *chicken salad* in the sentence "I love tuna sandwich and chicken salad very much" are opinion targets. We can also predict that the phrase *the design of iPhone 7* in the sentence "I love the design of iPhone 7"

RuleID	Rule	Example
R1	O $\xrightarrow{amod}$ T	They have nice dessert. (nice $\xrightarrow{amod}$ *dessert*)
R2	T $\xrightarrow{nsubj}$ O	Its camera is great. (*camera* $\xrightarrow{nsubj}$ great)
R3	T $\xrightarrow{dobj}$ O	I love their fries. (*fries* $\xrightarrow{dobj}$ love)
R4	T $\xrightarrow{nsubj}$ H $\xleftarrow{amod}$ O	iPhone is the best cell-phone. (*iPhone* $\xrightarrow{nsubj}$ phone $\xleftarrow{amod}$ best)
ER1	W $\xrightarrow{amod}$ T	I like Indian food. (*Indian* $\xrightarrow{amod}$ food)
ER2	W $\xrightarrow{nn}$ T	Their spring roll is great. (*spring* $\xrightarrow{nn}$ roll)
ER3	$W_2 \xrightarrow{pobj} W_1 \xrightarrow{prep}$ T	I like the design of iPhone. (*iPhone* $\xrightarrow{pobj}$ *of* $\xrightarrow{prep}$ design)

Table 4.1: Rules for detecting opinion targets. H represents any word. W represents an additional target word to be detected using the expansion rules.

is an opinion target. We can see that the two sentences come from very different domains, but the rule is general.

Based on the work by [77], we develop a set of rules that use syntactic patterns to detect potential opinion targets. The rules are based on three dependency relations: *amod*, *nsubj* and *dobj*. In the descriptions below, we use arrows to indicate the direction of the dependency relations. We use T to denote a potential opinion target and O to denote an opinion word. We use four rules (R1, R2, R3 and R4) shown in Table 4.1 to identify T.

In addition, we have the constraints that the opinion word O must come from a pre-defined sentiment lexicon, and the POS tag of the target T must be one of {NN, NNS, NNP, NNPS}.

The rules above can only help us identify the head word of an opinion target.

However, many opinion targets consist of multiple tokens. In order to identify additional tokens in opinion targets, we analyze the dependency relations within opinion targets and identify a set of expansion rules to expand opinion targets. They are shown as ER1, ER2 and ER3 in Table 4.1.

The rules described above can help us identify potential opinion targets in any given domain. However, using only these rules to extract opinion targets does not give very competitive results. This is because the coverage of these rules is still limited, and therefore the performance of a purely unsupervised method using these rules is not competitive, as we will see in the experiment section. In the next section, we will describe in detail how we combine these rules with recurrent neural network models to perform cross-domain opinion target extraction.

4.2.5 Two Architectures for Cross-domain Opinion Target Extraction

In this section, we will describe the neural network architectures we use to combine auxiliary labels with true labels in order to perform cross-domain opinion target extraction. The core of our models is to learn a hidden vector representation for each token that is useful for both the source and the target domains.

First of all, we have the following training data available to us. Recall that in the source domain, we have a set of sentences together with the true opinion target labels, denoted as $\mathcal{D}^s = \{(\mathbf{x}^s, \mathbf{y}^s)\}$. Next, for both sentences in the source domain and sentences in the target domain, based on the syntactic rules we have defined, we can obtain their auxiliary label sequences. Let us use $\mathbf{z}$ to denote the auxiliary labels of sentence $\mathbf{x}$. Let $\mathcal{D}^a = \{(\mathbf{x}^a, \mathbf{z}^a)\}$ denote all the sentences from the source and the target domains together with their auxiliary labels.

We now present two neural network architectures to use $\mathcal{D}^s$ and $\mathcal{D}^a$ to learn a prediction model. In both architectures, we introduce an auxiliary hidden layer $\mathbf{h}'$.

Concatenated Network

In the first architecture, we first use an RNN to create the auxiliary hidden layer $\mathbf{h}'$ as follows:

$$\mathbf{h}'_i = \mathit{RNN}_{\Theta'}(\mathbf{h}'_{i-1}, \mathbf{x}_i). \tag{4.5}$$

This hidden layer will be used to predict the auxiliary labels:

$$p(z_i \mid \mathbf{h}'_i) = \text{softmax}(\mathbf{W}'\mathbf{h}'_i + \mathbf{b}'). \tag{4.6}$$

We also use a different RNN to create the standard hidden layer $\mathbf{h}$ as follows:

$$\mathbf{h}_i = \mathit{RNN}_{\Theta}(\mathbf{h}_{i-1}, \mathbf{x}_i). \tag{4.7}$$

Next, we concatenate $\mathbf{h}$ and $\mathbf{h}'$ into a single vector:

$$\overline{\mathbf{h}}_i = \mathbf{h}_i \bigoplus \mathbf{h}'_i. \tag{4.8}$$

This concatenated hidden vector is then used to predict the true opinion target label:

$$p(y_i \mid \overline{\mathbf{h}}_i) = \text{softmax}(\mathbf{W}\overline{\mathbf{h}}_i + \mathbf{b}). \tag{4.9}$$

This architecture is shown in Figure 4.2(b). We can see that different from a standard model, this model uses the additional auxiliary hidden vector $\mathbf{h}'$ together with $\mathbf{h}$ to predict the final labels.

Hierarchical Network

In the second architecture, the auxiliary hidden vector $\mathbf{h}'$ is defined in the same way as in Eqn. (4.5), and the probability distribution $p(z_i \mid \mathbf{h}'_i)$ is also defined in the same

way as in Eqn. (4.6). However, the standard hidden layer $\mathbf{h}$ now uses $\mathbf{h}'$ as input:

$$\mathbf{h}_i = \mathit{RNN}_\Theta(\mathbf{h}_{i-1}, \mathbf{h}'_i). \tag{4.10}$$

And finally, to predict the true opinion target label, we have

$$p(y_i \mid \mathbf{h}_i) = \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{b}). \tag{4.11}$$

This architecture is shown in Figure 4.2(c). We call this the hierarchical network because $\mathbf{h}'$ and $\mathbf{h}$ now reside at different layers of the neural network.

While both architectures can combine the power of domain-independent rules and the true opinion target labels from the source domain, they differ in how knowledge from these two parts are integrated. In the Concatenated Network, there is not much interaction between $\mathbf{h}'$ and $\mathbf{y}$. The Hierarchical Network has a more complicated mechanism by feeding $\mathbf{h}'$ into the RNN that produces $\mathbf{h}$. However, both models share the similar idea of (1) using auxiliary labels to encode domain-independent rules, and (2) learning parameters based on true annotated labels and auxiliary labels to obtain representation vectors that are useful across different domains.

4.2.6 Learning the Parameters

To learn the parameters, we use the commonly-used log likelihood objective function. Note that there are two parts in our loss function, one related to the auxiliary labels $\mathbf{z}$ and the other related to the true labels $\mathbf{y}$.

Let us define the following loss functions:

$$L_z = \sum_{(\mathbf{x}^a, \mathbf{z}^a) \in \mathcal{D}^a} -\log p(\mathbf{z}^a \mid \mathbf{x}^a), L_y = \sum_{(\mathbf{x}^s, \mathbf{y}^s) \in \mathcal{D}^s} -\log p(\mathbf{y}^s \mid \mathbf{x}^s). \tag{4.12}$$

To learn the parameters of our model, we can divide our training process into two steps by first minimizing L_z and then minimizing L_y with all the parameters

Dataset	# Sentences	# Words
Restaurant	5,841	88,707
Laptop	3,845	63,011
Device	3,836	70,913
Service	8,545	159,742

Table 4.2: Basic statistics of the datasets.

learned by minimizing L_z fixed. We can also jointly minimize the sum of L_z and L_y with respect to all the parameters. We refer to the former as *separate* training and the latter as *joint* training. Back propagation is used in both training strategies. In our experiments, we will compare their performance.

4.3 Experiments

Datasets

We use reviews from four different domains for our experiments. The four domains are restaurant, laptop, digital device and web service. The restaurant data is a combination of the restaurant reviews from SemEval 2014 [76] and SemEval 2015 [75]. The laptop data comes from SemEval 2015 [75]. The digital device dataset contains review sentences on five digital devices and was created by [41]. The web service dataset was introduced by [88] and consists of sentences from reviews of web services. The sentiment lexicon we use was downloaded from University of Illinois at Chicago.[1]

During preprocessing, all words are converted to lowercase, URL links are replaced with <URL> and numbers are replaced with <NUM>. We also remove some noisy sentences in the web service dataset. After preprocessing, the basic statistics of our datasets are shown in Table 4.2. For simplicity, we use Restaurant, Laptop, Device and Service to denote each of the datasets, respectively.

[1] https://www.cs.uic.edu/∼liub/FBS/sentiment-analysis.html#lexicon

Evaluation Metric

We use the $F1$ score of opinion targets as the evaluation metric. Following previous work [51, 101], we only consider exact matches, which means a target is considered correctly extracted only if the output of a model is exactly the same without any missing word or extra word.

Parameter Settings

We use pre-trained word embeddings from Google word2vec[2] to initialize the word embeddings in our models. The word embeddings are updated in the training process. To learn the parameters of our model, we use the Adagrad algorithm with a mini-batch size of 10 sentences. The initial learning rate is set to 0.01. Following previous work [51, 93, 101], we concatenate the word embeddings of the current word, its previous word and next word as the input to our model. This setting is also used in our neural network baselines. For each target domain, we leave out 200 randomly selected sentences as validation set. The dimension of the hidden layers is determined according to the performance on the validation set. We find 100 to be the best and the reported results below are obtained using 100 as the hidden layer dimension. All neural network models are trained for 15 iterations. Based on performance on the validation set, we choose the best model across the 15 iterations as our final model.

4.3.1 Models for Comparison

We use the following baselines for comparison:

- **CRF**: This is a traditional sequence labeling model using Conditional Random Field and discrete features such as word types, POS tags and dependency relations. It has been used for both single-domain and cross-domain opinion target extraction [33].

[2]https://code.google.com/archive/p/word2vec/

- **mDA**: This is a recently proposed domain adaptation method using marginalized denoising auto-encoders [9]. We use features that have been proven to be useful for opinion target extraction, including current word, previous word, next word, current POS tag, and the shortest dependency path and distance to an opinion word. The features are largely the same as in the **CRF** method.

- **FEMA**: This is another recently proposed domain adaptation method based on cross-domain feature embeddings [97]. We use the same feature templates as those for mDA.

- **Direct-1**, **Direct-2**: These two methods naively use the labeled training data from the source domain to train an LSTM model and apply it to the target domain test data. **Direct-2** uses two layers of LSTM. This is to compare with our hierarchical model, which also uses two layers of LSTM.

- **Aux**: This is an unsupervised method where we directly use the rules presented earlier to extract opinion targets.

- **Direct-Aux**: This is a naive way of using both the labeled training data from the source domain and the general syntactic rules. Essentially, we train an LSTM model using the combination of the source domain data with the true labels and the target domain data with the auxiliary labels generated by the rules.

Meanwhile, we have the following variants of our proposed models:

- **Con-Sep** The concatenated network trained with separate training.

- **Con-Joint** The concatenated network trained with joint training.

- **Hier-Sep** The hierarchical network trained with separate training. After the parameters Θ' are learned and the word embeddings updated, they are kept fixed. The subsequent training only updates Θ in Eqn. (4.7) and $\mathbf{W}$ and $\mathbf{b}$ in Eqn. (4.9).

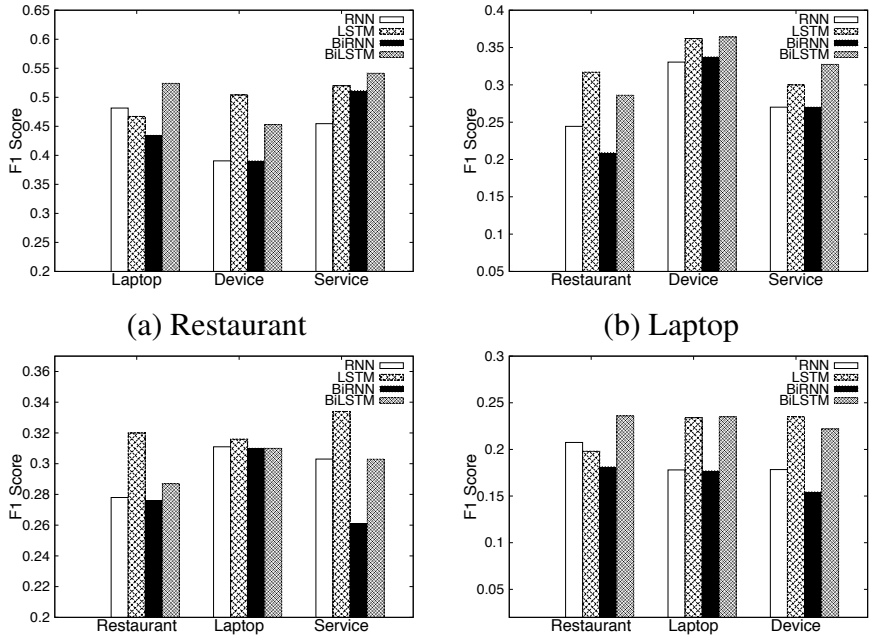
Data	CRF	mDA	FEMA	Direct-1	Direct-2	Aux	Direct-Aux	Con-Sep	Con-Joint	Hier-Sep	Hier-Joint
L-R	0.170	0.243	0.350	0.301	0.364	0.390	0.436	0.302	0.338	0.436	**0.467**†
D-R	0.025	0.213	0.207	0.306	0.352	0.390	0.432	0.280	0.281	0.451†	**0.504**†
S-R	0.170	0.325	0.376	0.439	0.458	0.390	0.434	0.458	0.421	0.479†	**0.520**†
R-L	0.109	0.209	0.266	0.277	0.290	0.199	0.210	0.248	0.292	0.269	**0.317**†
D-L	0.245	0.257	0.268	0.323	0.353	0.199	0.197	0.334	0.329	0.252	**0.362**†
S-L	0.116	0.146	0.150	0.252	0.249	0.199	0.219	0.256	0.288†	0.260	**0.300**†
R-D	0.090	0.172	0.229	0.246	0.225	0.233	0.246	0.269†	0.295†	0.300†	**0.320**†
L-D	0.270	0.294	0.296	0.296	0.300	0.233	0.226	0.308	0.316	0.279	**0.316**
S-D	0.097	0.169	0.187	0.283	0.275	0.233	0.241	0.243	0.283	0.309†	**0.334**†
R-S	0.088	0.131	0.108	0.146	0.175	0.151	0.127	0.167	0.145	0.180†	**0.198**†
L-S	0.086	0.131	0.148	0.152	0.183	0.151	0.126	0.150	0.185†	0.111	**0.234**†
D-S	0.045	0.095	0.088	0.165	0.151	0.151	0.132	0.166	0.186†	**0.241**†	0.235†
Average	0.126	0.199	0.223	0.265	0.281	0.243	0.252	0.265	0.280	0.297	**0.342**

Table 4.3: F1 scores achieved by the various methods we consider. The *Data* column shows the source and the target domains, where *L* stands for laptop, *R* stands for restaurant, *D* stands for device and *S* stands for service. † indicates that the result is statistically significantly better than CRF, mDA, FEMA, Direct-1, Direct-2, Aux and Direct-Aux with $p < 0.01$ based on McNemar's test. As an upper bound, we note that the F1 scores on the four domains *R*, *L*, *D* and *S* when trained on in-domain data are 0.779, 0.766, 0.451 and 0.438, respectively.

- **Hier-Joint** The hierarchical network trained with joint training.

4.3.2 Experiment Results

We show the $F1$ scores of the various methods under different source-target domain settings in Table 4.3. From the table we can observe the following: (1) Our proposed model with the hierarchical network and joint learning can outperform all the other methods under all settings except one. Furthermore, most of the time the improvement is statistically significant. This shows the advantage of our proposed model with the Hier-Joint setting. (2) Our proposed model with the other settings (Con-Sep, Con-Joint and Hier-Sep) also tend to work well in many cases, outperforming the baselines. This shows that in general our idea of learning a hidden representation using the auxiliary labels is effective. (3) Comparison among Con-Sep, Con-Joint, Hier-Sep and Hier-Joint shows that using joint learning helps and using the hierarchical network generally is better than using the concatenated network. (4) Among the baselines, Direct-2 tends to work well in general except when



(a) Restaurant (b) Laptop

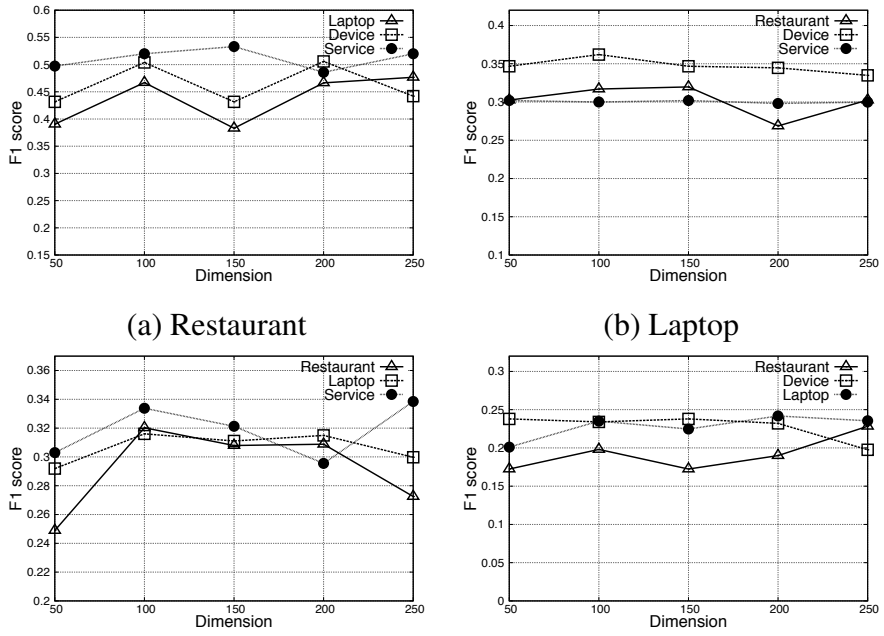
(c) Device (d) Service

Figure 4.3: Effect of RNN unit.

Restaurant is the target domain, in which case Aux and Direct+Aux generally work better. This shows that none of the baselines we consider is guaranteed to work well in a cross-domain setting for the opinion target extraction task.

Overall, the results demonstrate that our hierarchical network with joint learning can integrate labeled dataset from the source domain with domain-independent syntactic rules well. The reinforcement between these two types of information makes this model more effective than other models for cross-domain opinion target extraction.

To understand how our Hier-Joint model obtains better performance over the others, we compare the precision and recall of Hier-Joint with all the other baselines. We find that Hier-Joint can get both better precision and better recall most of the time. It demonstrates that our hierarchical network with auxiliary labels can discover more targets without bringing in many false positive predictions.



(a) Restaurant (b) Laptop

(c) Device (d) Service

Figure 4.4: Effect of hidden dimensions.

Comparison of different RNNs

The results above are based on LSTM, which is a special case of RNN. To compare the effectiveness of different types of RNNs, we use each dataset as the target domain and compare the performances of different RNNs. We consider standard RNN, Bi(bi-directional)RNN, LSTM and Bi(bi-directional)LSTM. The results are shown in Figure 4.3 with X-axis showing the source domain. We can see that LSTM and BiLSTM consistently achieve better $F1$ scores than RNN and BiRNN over all source-target domain pairs. However, there is no clear winner between LSTM and BiLSTM

Effect of the hidden layer dimension

We also study the effect of using different hidden layer dimensions. We show the results in Figure 4.4. The results are from the Hier-Joint model, which is the best among our proposed models. We can see that the $F1$ score first goes up. For most target domains, it reaches the optimal value at dimension 100 or 150 and then starts to go down. One possible reason for the drop in $F1$ score is that the model starts to

Sentence	Hier-Joint	Direct	Aux
But **dinner** here is never disappointing , even if the **prices** are a bit over the top .	dinner prices	prices	dinner
All my co-workers were amazed at how small the **dish** was .	dish	co-workers dish	co-workers
The **noise level** was unbearable , conversation impossible .	noise level	*NONE*	noise level conversation
My friend got the **mushroom pizza** which tasted better .	mushroom pizza	*NONE*	*NONE*

Table 4.4: Sampled sentences and the targets extracted by different models. We use Service as source domain and restaurant as target domain. The gold standard targets are highlighted in black font.

overfit the training data when the hidden layer dimension becomes too high.

Case Study

To understand how Hier-Joint works better than Direct and Aux, we sample 4 sentences from Restaurant domain and show the extracted targets of Hier-Joint, Direct and Aux when using Service as the source domain in Table 4.4. Direct fails to identify domain specific targets, like *dinner* and *mushroom pizza*. Aux can only extract those that have certain relations with sentiment words, like *noise level* and *dinner*. We can see that Direct and Aux are actually complementary to each other. As an integration of these two methods, Hier-Joint can successfully extract all the targets in these sentences.

4.4 Discussion

In this chapter, we propose two RNN-based neural networks for cross-domain opinion target extraction. We first use unsupervised syntactic rules to generate an auxiliary label sequence for each sentence. We then train our models using both the true labels and the auxiliary labels. By leveraging knowledge from labeled training data and domain-independent syntactic rules at the same time, our Hierarchical Network with joint learning can learn a robust vector representation that is useful

across domains and outperform several strong baselines. This chapter shows that it is a promising direction to boost RNNs with rules and auxiliary tasks for opinion target extraction.

Chapter 5

Modelling Product Reviews with Word Vectors

The third step for aspect discovery from product reviews is to cluster words into groups according to the aspects they are referring to. Although various models have been proposed for aspect word clustering, they mostly rely on lexical similarity to calculate textual similarity. However, as reviews are edited freely by customers, a diverse range of words may be used. This renders the traditional ways of calculating textual similarity ineffective. In this chapter, we apply vector representation of words to measure the semantic similarity between text. We design a model that seamlessly integrates word vectors into a joint model of user feedback and text content. Extensive experiments on datasets from various domains prove that our model is effective in both aspect discovery and recommendation in social media.

5.1 Introduction

With the explosive usage of online shopping websites and reviews websites, online reviews are accumulating and becoming more and more important to various tasks. They can help us understand users' online behaviour and accurately model their preferences and interests. One important applications of using reviews is to recom-

mend product to potential customers. Indeed there has been several recent studies trying to combine textual data with rating or adoption data for recommendation. Among these studies, many use textual information separately from their recommendation model. They first extract useful information from text and then embed such information in their recommendation models [8, 91]. Some other work uses a unified, principled model to combine text with rating or adoption data [1, 49, 57].

A limitation of these recent studies is that their textual similarity is based on lexical similarity only. When two items' descriptions are semantically related but use different words, these models may not consider the two items to be similar. In online platforms, however, the vocabulary used is very diverse and two pieces of text can be semantically similar even with low lexical overlap, so semantic similarity is especially important when we analyze social media content. Assume that we know that product A is an excellent gift for mothers and product B is often bought for people's parents from their online reviews. Based on the semantic similarity between "mother" and "parent", we can be confident to recommend B to a user who has bought A or shows interest in buying a gift for his mother even if few people have bought both of them and their reviews have few overlapping words. What's more, in the Meetup dataset we use, which is about online interest groups and organized events, there is a group tagged with "Buddhism" and another group tagged with "vegetarian." If we only consider lexical similarity, these two groups may not be considered related based on the tags. However, we should probably recommend the second group to users who have joined the first group as many Buddhists are also vegetarians. The challenge is how to incorporate the consideration of semantic similarity based on textual descriptions into a traditional collaborative filtering framework in a principled way.

With the recent advances in learning word embeddings from large corpora, we can use vector representation of words to measure the semantic similarity between two pieces of text. Word embeddings are techniques that can project words into vectors carrying their semantic meanings [59, 73]. In this study, we propose a new

document modelling technique that makes use of word embeddings and integrate it with collaborative filtering for better product recommendation. Our model can jointly model ratings, latent factors, topics and word embedding vectors simultaneously. With the help of vector representations of words, the model is able to learn cleaner topics, more accurate latent factors and provide better recommendations. Extensive experiments show that our model outperforms other methods on item recommendation and topic discovery. For example, for the Meetup data, our method can successfully recommend the "vegetarian" group to users who have joined the "Buddhism" group, and based on the ground truth, for such users our method indeed gives better performance than other baseline methods we consider.

5.2 Method

In this section, we formally formulate our problem and present our proposed model. Based on our model's properties, we denote it by **C**ollaborative filtering with word **E**mbedding-based **T**opic models (CET). We will first give a brief description of the background the notations. Then the model and the parameter estimation algorithm are formally described.

5.2.1 Problem Formulation and Notation

Suppose we have a collection of N_I items $\mathcal{I} = \{i_1, i_2, \cdots, i_{N_I}\}$ and a collection of N_U users $\mathcal{U} = \{u_1, u_2, \cdots, u_{N_U}\}$. We also observe a collection of ratings[1] $\mathcal{R} = \{r_{ui}\}$ where r_{ui} is the rating of item i by user u. For each item, there is an associated document d_i, which is a sequence of words. This document can be from different sources of user-generated content. For example, in online review websites, we can use the reviews of a product as the document associated with the product. For items that have user-assigned tags, we can use the set of tags as the associated document

[1]For convenience, we assume we have numerical rating data, but the model can be easily generalized for binary adoption data.

Variable	Description
r_{ui}	Rating of item i by user u
$\boldsymbol{v}_w, \boldsymbol{v}_{wi}$	Vector of word w learned by word embeddings and corresponding value at the ith dimension
$\boldsymbol{\alpha}$	The hyper-parameters for the Dirichlet distribution
$\sigma_U, \sigma_I, \sigma_R$	The standard deviation for univariate Gaussian distributions
$\boldsymbol{\Gamma}_U, \boldsymbol{\Gamma}_I$	The covariance matrices for multivariate Gaussian distribution
b_u, b_i	The rating bias of user u and rating bias of item i
$\boldsymbol{p}_u, \boldsymbol{q}_i$	The latent factor of user u and latent factor of item i
$\boldsymbol{\theta}_i$	The topic distribution of item i
$\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$	The mean and covariance matrix for the multivariate Gaussian distribution of topic t
Dir($\boldsymbol{\alpha}$)	A Dirichlet distribution with hyper-parameter $\boldsymbol{\alpha}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	A Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
Multi($\boldsymbol{\theta}$)	A discrete distribution with $\boldsymbol{\theta}$ as parameter

Table 5.1: Notation of our model.

for an item. The set of all words appearing in our data comprises the vocabulary $\mathcal{V}$ and for each word w of this vocabulary, we assume that we have a pre-trained vector v_w of dimension K, which can be learned by word embedding models [59, 73]. Our task is to recommend items to users according to both their rating histories and the textual data generated by users in social media.

5.2.2 Collaborative Filtering with Word Embedding-based Topic Models

Our model is based on matrix factorization, topic modeling and word embedding vectors. On the rating part, we apply matrix factorization as the generative process. On the text part, we design a generative process based on Latent Dirichlet Allocation (LDA) [2]. We also assume that the topic distribution of an item is linked to the item's latent vector used in matrix factorization, which is an idea previously explored in [49, 57, 90]. By doing this, we build a single unified and principled model that combines text and ratings. Similar to [90], we assume that item factors are derived from the corresponding topic distributions instead of setting them to be

identical. Specifically, item i's latent factor $\boldsymbol{q}_i$ is sampled from a multivariate normal distribution with its topic distribution $\boldsymbol{\theta}_i$, which is derived from review text, as the mean:

$$\boldsymbol{q}_i \sim \mathcal{N}(\boldsymbol{\theta}_i, \boldsymbol{\Gamma}_I) \tag{5.1}$$

where $\boldsymbol{\Gamma}_I$ represents the covariance matrix. This renders our model more flexible in modeling latent factors. Different from standard LDA, which treats each word as a single discrete symbol, we use the vector representations of them instead. We still assume that there is a multinomial topic distribution for each document. But for each topic, we assume there is a multivariate Gaussian distribution, which is used to generate word vectors. There are two parameters for each topic t, which are the mean vector $\boldsymbol{\mu}_t$ and co-variance matrix $\boldsymbol{\Sigma}_t$. To generate a word in a document, we first need to sample a topic according to the document-topic distribution, and then sample a vector from the Gaussian distribution of the sampled topic. The generative process of our model is shown below and we list the used notation in Table 5.1.

- For each user, sample a bias $b_u \sim \mathcal{N}(0, \sigma_U)$ and a latent vector $\boldsymbol{p}_u \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Gamma}_U)$.
- For each item i, sample a topic distribution $\boldsymbol{\theta}_i \sim \text{Dir}(\boldsymbol{\alpha})$ for text. Sample a latent vector $\boldsymbol{q}_i \sim \mathcal{N}(\boldsymbol{\theta}_i, \boldsymbol{\Gamma}_I)$ and a bias $b_i \sim \mathcal{N}(0, \sigma_I)$. For each word w in the associated text:
  - Sample a topic $z \sim \text{Multi}(\boldsymbol{\theta}_i)$.
  - Sample a word embedding vector $v_w \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$.
- For rating of item i by user u, sample a numerical value $r_{ui} \sim \mathcal{N}(b_u + b_i + \boldsymbol{p}_u^{\intercal}\boldsymbol{q}_i, \sigma_R)$.

With this model, we can find the underlying topics of words based on their semantic meanings. This can help us recommend items to users even if the used text is very diverse, which is common in social media. For example, although "fitness" and

"exercise" are two different words, in pre-trained word embeddings, their distance is smaller than a random pair of words, so they are more likely to be generated by the same multivariate Gaussian than from different Gaussian distributions. In our model, items whose descriptions contain "fitness" and items whose descriptions contain "exercise" will have similar topic distributions and so are their latent factors. Then for a user who has adopted items with the word "fitness," our model is more likely to recommend items with the word "exercise" to him. Unfortunately, traditional models may not achieve this as they do not consider the semantic meaning of words.

It is worth pointing out that in our CET model, the modified LDA component, which generates word embedding vectors from a mixture of multivariate Gaussian distributions, is essentially the same as in a recent work by Das et. al. [17]. Both models assume that each document has a multinomial topic distribution with a Dirichlet distribution as the prior. Both model also assume that there is a multivariate Gaussian distribution for each topic and word vectors are sampled from these Gaussian distributions. The only difference is that in Gaussian-LDA, the hyper-parameter α, which is used in the prior Dirichlet distribution, is to be learned. But in our model, we assume that it is a parameter set by users. However, we developed our model independently and our focus is to apply the model for the purpose of recommendation. Note also that although here we assume the text is associated with each item, our model is not restricted to this setting. If there is text associated with users, our model can also be directly applied by switching the generative process of items with that of users.

5.2.3 Parameter Estimation

When applying our model to a dataset, text, ratings and word vectors are all given, and we need to find the hidden parameters that can maximize the posterior likelihood. So, our goal of training is to learn the parameters that can maximize the

following probability:

$$\begin{aligned} & P(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}_U, \boldsymbol{B}_I, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma} | \boldsymbol{W}, \boldsymbol{R}) \\ \propto \; & P(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}_U, \boldsymbol{B}_I, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{W}, \boldsymbol{R}) \\ = \; & P(\boldsymbol{P}) P(\boldsymbol{Q}|\boldsymbol{\theta}) P(\boldsymbol{B}_U) P(\boldsymbol{B}_I) P(\boldsymbol{\theta}) P(\boldsymbol{\mu}) P(\boldsymbol{\Sigma}) P(\boldsymbol{W}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ & P(\boldsymbol{R}|\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}_U, \boldsymbol{B}_I) \end{aligned} \tag{5.2}$$

where

$$P(\boldsymbol{p}_u) = \mathcal{N}(\boldsymbol{p}_u|0, \sigma_U), P(\boldsymbol{q}_i) = \mathcal{N}(\boldsymbol{q}_i|\boldsymbol{\theta}_i, \sigma_I) \tag{5.3}$$

$$P(b_u) = \mathcal{N}(b_u|0, \sigma_U), P(b_i) = \mathcal{N}(b_i|0, \sigma_I) \tag{5.4}$$

$$P(\boldsymbol{\theta}_i) = \text{Dir}(\boldsymbol{\theta}_i|\alpha) \tag{5.5}$$

and

$$P(\boldsymbol{W}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_i \prod_{v \in d_i} N(\boldsymbol{w}_v|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{5.6}$$

$$P(\boldsymbol{R}|\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}_U, \boldsymbol{B}_I) = \prod_{\{u,i\} \in \mathcal{D}} \mathcal{N}(r_{ui}|\boldsymbol{p}_u^\intercal \boldsymbol{q}_i + b_u + b_i, \sigma_R) \tag{5.7}$$

Here $\boldsymbol{P}$ and $\boldsymbol{Q}$ refer to all latent vectors for items and users, $\boldsymbol{B}_U$ and $\boldsymbol{B}_I$ refer to bias terms of users and items. $\boldsymbol{W}$ refers to all the words we observe and $\boldsymbol{R}$ refers to all the ratings. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ represent all means and covariance matrices of the Gaussian distributions of all topics. As there is no closed form solution for our problem, we use Gibbs-EM algorithm [89] for parameter estimation. For each iteration, we alternate between Gibbs sampling and gradient descent. More specifically, in each iteration, we first perform Gibbs sampling based on parameters learned in the last iteration, which will be fixed in the sampling stage. Then based on the sampled hidden variables, we optimize our objective function using gradient descent.

E-step: We fix the parameters $\boldsymbol{\theta}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and collect samples of the hidden variables $\boldsymbol{Z}$ to approximate the distribution $P(\boldsymbol{Z}|\boldsymbol{W}, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The distribution of

the hidden labels for Gibbs Sampling is:

$$P(z_{ij} = t) \propto \boldsymbol{\theta}_{it} \cdot \mathcal{N}(\boldsymbol{v}_{w_{ij}} | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \tag{5.8}$$

Here, z_{ij} is the topic assignment of the word at the jth position of text of item i and w_{ij} denotes the corresponding word.

M-step: With the collected samples of $\boldsymbol{Z}$, we need to find the values of $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{B}_U$, $\boldsymbol{B}_I$, $\boldsymbol{\theta}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ that maximize the following objective function:

$$\mathcal{L} = \sum_{\boldsymbol{Z} \in \mathcal{S}} \log P(\boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{R}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}_U, \boldsymbol{B}_I, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma} | \alpha, \sigma_U, \sigma_I, \sigma_R, \Gamma_U, \Gamma_I), \tag{5.9}$$

where $\mathcal{S}$ is the set of samples collected in the E-step. The full expansion of the objective function is:

$$\begin{aligned}
\mathcal{L} &= \sum_{\boldsymbol{Z} \in \mathcal{S}} \log P(\boldsymbol{W} | \boldsymbol{Z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) P(\boldsymbol{Z} | \boldsymbol{\theta}) P(\boldsymbol{R} | \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}_U, \boldsymbol{B}_I) \cdot \\
&\qquad P(\boldsymbol{\theta} | \alpha) P(\boldsymbol{P} | \Gamma_U) P(\boldsymbol{Q} | \boldsymbol{\theta}, \Gamma_I) P(\boldsymbol{B}_U | \sigma_U) P(\boldsymbol{B}_I | \sigma_I) \\
&= \sum_{\boldsymbol{Z} \in \mathcal{S}} \{ \sum_{i=1}^{N_I} \sum_{j=1}^{L_i} \log \mathcal{N}(\boldsymbol{v}_{w_{ij}} | \boldsymbol{\mu}_{z_{ij}}, \boldsymbol{\Sigma}_{z_{ij}}) \boldsymbol{\theta}_{i z_{ij}} \\
&\quad + \sum_{j=1}^{N} \log \mathcal{N}(r_j | \boldsymbol{p}_u^\intercal \boldsymbol{q}_i + b_u + b_i) + \sum_{d=1}^{D} \log \mathrm{Dir}(\boldsymbol{\theta}_d | \boldsymbol{\alpha}) \\
&\quad + \sum_{i=1}^{I} \log \mathcal{N}(\boldsymbol{q}_i | \boldsymbol{\theta}_i, \boldsymbol{\Gamma}_I) + \sum_{u=1}^{U} \log \mathcal{N}(\boldsymbol{p}_u | 0, \boldsymbol{\Gamma}_U) \\
&\quad + \sum_{i=1}^{I} \log \mathcal{N}(b_i | 0, \sigma_I) + \sum_{u=1}^{U} \log \mathcal{N}(b_u | 0, \sigma_U) \}.
\end{aligned} \tag{5.10}$$

It is noted that θ for any document is constrained to be a multinomial distribution. To transform this constrained optimization problem to an unconstraint one, we use a set of auxiliary variables λ_{it} to replace θ_{it} with $\frac{\exp(\lambda_{it})}{\sum_{t'} \exp(\lambda_{it'})}$. We use gradient descent to find the optimal value of $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{B}_U$, $\boldsymbol{B}_I$, $\boldsymbol{\theta}$. Based on the expanded

objective function, the update formulas are:

$$\lambda_{it} = \lambda_{it} + \gamma \cdot \{ \sum_{k=1,k\neq t}^{T} [\frac{1}{\mathbf{\Gamma}_I^{kk}}(\boldsymbol{\theta}_{ik} - \boldsymbol{q}_{ik})\boldsymbol{\theta}_{ik}\boldsymbol{\theta}_{it} + (N_{ik} + \alpha - 1)\boldsymbol{\theta}_{it}]$$
$$+ \frac{1}{\mathbf{\Gamma}_I^{tt}}(\boldsymbol{q}_{it} - \boldsymbol{\theta}_{it})(\boldsymbol{\theta}_{it} - \boldsymbol{\theta}_{it}^2) + (N_{it} + \alpha - 1)(1 - \boldsymbol{\theta}_{it})\} \tag{5.11}$$

$$\boldsymbol{q}_{it} = \boldsymbol{q}_{it} + \gamma \cdot [- \sum_{j \in R_I^i} \frac{1}{\sigma_R} e_j \boldsymbol{p}_{ut} + \frac{1}{\mathbf{\Gamma}_I^{tt}}(\boldsymbol{\theta}_{it} - \boldsymbol{q}_{it})] \tag{5.12}$$

$$\boldsymbol{p}_{ut} = \boldsymbol{p}_{ut} + \gamma \cdot (- \sum_{j \in R_U^u} \frac{1}{\sigma_R} e_j \boldsymbol{q}_{it} - \frac{1}{\mathbf{\Gamma}_U^{tt}} \boldsymbol{p}_{ut}) \tag{5.13}$$

$$b_i = b_i + \gamma \cdot (- \sum_{j \in R_I^i} \frac{1}{\sigma_R} e_j - \frac{1}{\sigma_I} b_i) \tag{5.14}$$

$$b_u = b_u + \gamma \cdot (- \sum_{j \in R_U^u} \frac{1}{\sigma_R} e_j - \frac{1}{\sigma_U} b_u). \tag{5.15}$$

$\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ can be updated using the following equations:

$$\boldsymbol{\mu}_{ti} = \frac{1}{N_{t\cdot}} \sum_{w=1}^{V} N_{tw} \boldsymbol{v}_{wi} \qquad \mathbf{\Sigma}_t^{ii} = \frac{1}{N_{t\cdot}} \sum_{w=1}^{V} N_{tw} (\boldsymbol{v}_{wi} - \boldsymbol{\mu}_{ti})^2. \tag{5.16}$$

where N_{tv} is the number of times word type w is assigned to topic t, $N_{t\cdot}$ is the number of times all word types are assigned to topic t and $\mathbf{\Sigma}_t^{ii}$ is the element at row i, column i of matrix $\mathbf{\Sigma}_t$.

After all parameters in the model are learned, we use $\hat{r}_{ui} = \boldsymbol{p}_u^{\intercal} \boldsymbol{q}_i + b_u + b_i$ to predict the rating of item i by user u. In our implementation, we perform 600 runs of Gibbs EM. Because Gibbs sampling is time consuming, in each run we only perform one iteration of Gibbs sampling and collect that one sample. We then have 60 iterations of gradient descent. The gradient descent algorithm we use is L-BFGS, which is efficient for large scale data set [68]. We downloaded word vectors from the homepage of word2vector[2] and use them as our word embedding vectors.

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

5.3 Experiment

Our model can be applied to many recommendation tasks on social media where user-generated content plays an important role. To test it, we pick two representative social media platforms for experiments. The first is Meetup[3], an event-based online social network. Meetup allows users to create interest groups and organize events. A commonly studied recommendation task on Meetup is how to recommend an interest group to a user. The second is Amazon's product review platform. We use user-generated product reviews as additional textual information to help product recommendation. The content in these two platforms are also representative. In the online social network website we use, content contains tags given by users. Because there is not a controlled vocabulary of tags and the number of tags assigned to each item can be small, the data is very sparse. In online review website, users can write their reviews in free form. So the content is relatively rich but the diversity is still high.

For each dataset, we use 10% of the data as the development set and another 10% of the data as the testing set. The remaining 80% of the data is used for training. We tune all models according to the development set and test them on the testing set. As our model does not update word embedding vectors. Those words with no pre-trained vectors are of no use to CET. So we just delete them all. The average percentage of words with embedding vectors is 54.7% over all datasets. To show the effectiveness of our model, we choose several appropriate state-of-the-art recommendation techniques for comparison. Besides showing their performance, we also do statistical significance test of results using Wilcoxon signed-rank test.

5.3.1 Group Recommendation in Meetup

The first experiment is conducted on a Meetup dataset [52]. Meetup is an online event-based social network. In this website, users can build or join groups and each

[3] `http://www.meetup.com`

group can organize and publish offline events for people to participate in. Users and groups can use tags to label themselves to show their interests. The text we use is tags associated with groups. As some of these tags are phrases with multiple words, we add up the embedding vectors of all words in the tag to get the embedding vector of the tag. The dataset we use is a random sample from the data used in [52]. There are 2225 users, 6950 groups, 8015 user-group membership pairs and each group has 7.06 tags on average. This data is very sparse as only 0.04% of its user-group matrix entries contain values. For this dataset, we only have the information about which groups a user has joined. For the groups the user has not joined, there can be different reasons. The user may not like the group or the user may be unaware of the group at all. This type of negative examples is called implicit feedback. Because of this, we choose two models that work on implicit feedback as our baselines as follows.

CTR: Collaborative Topic Regression [90] is a model designed for scientific article recommendation with implicit feedback. It assumes that each article's latent factor is a deviation from its topic distribution.

OCF: One-class Collaborative Filtering [70] extends traditional matrix factorization to model implicit feedback. In our experiments, we use the re-weighting technique proposed in this paper.

Quantitative Study

We use MPR (Mean Percentage Ranking) [28] as the evaluation metric. For each user-group pair in our testing data, we randomly select 1,000 "negative" groups and mix them with the "positive" group. We rank all these 1,001 groups based the predicted rating from the target user. Then, we calculate MPR as follows:

$$\text{MPR} = \frac{1}{N}\sum_{i=1}^{N}\frac{R_i}{M}, \tag{5.17}$$

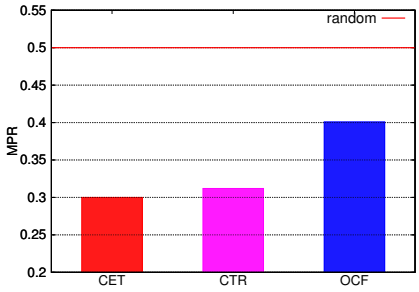


Figure 5.1: Mean Average Ranking (MAR) for CET, CTR and OCF on Meetup data.

where R_i is the position of the adopted group in testing pair i and M is the number of ranked groups, which is 1001 in our experiment, and N is the number of pairs in testing data. For a testing instance i, Percentage Ranking (PR) is defined as $\frac{R_i}{M}$, which will be used in the next subsection.

The MPR for CET, CTR and OCF are shown in Figure 5.1. OCF performs the worst for this dataset. This is because the dataset is too sparse and it is very hard to learn useful item latent vectors purely based on user membership information. By utilizing tag information, CTR can obtain a much better MPR value. CET can even outperform CTR by using word embedding vectors as it utilize the semantic meaning of words. Statistical test shows that CET's performance is significantly better than CTR and OCF at 5% level. It proves that compared with the baselines, our model can learn latent factors much more effectively.

Qualitative Study

To qualitatively understand how our model outperforms CTR, we display some sample users in Table 5.2. The representative tags of groups they have joined as indicated in the training data, the tags of "positive" groups in the testing data (i.e. groups that should be recommended) and the corresponding Percentage Ranking (PR) by CET and CTR are also shown together. For user 1408, the tags of groups he has joined tell us that he is interested in exercises and outdoor activities. A recommendation method should rank groups related to this topic higher than others. The group

User ID	Tags of groups they have joined	Tags of groups we should recommend	PR by CET	PR by CTR
1408	fitnees friends music meditation hiking yoga	aerobics running	0.020	0.577
1247	cooking nutrition movies fitness	volleyball	0.135	0.663
835	photo weightloss fitness	theater art museum	0.001	0.528
399	photoshop alternative mediation buddhism	vegetarian nutrition	0.042	0.563

Table 5.2: Sampled users, the representative tags of groups they join, the tags of group we should recommend and the percentage ranking of CET and CTR.

with tags "aerobics" and "running" shows up in our test instances. Our CET model ranks it higher than 98% of the negative examples while CTR only ranks it higher than 42% of the negative examples. The reason is that tags used in social media is very diverse, and groups with similar properties may share no words at all. Traditional way of using lexical similarity to compute textual similarity cannot work very well in this case. So it becomes hard for them to recommend groups based on tags. However, by leveraging words' vector representation, CET can tackle this problem better. The second and third cases also prove this. It is interesting that CET is also able to recommend groups that is conceptually related but have different properties. In the fourth row, we can see that user 399 is interested in Buddhism, and he has also joined a group about vegetarian, which appears in our test dataset. Buddhism is about religion while vegetarian is about food preference. It is impossible to connect them based only on lexical similarity. However, we know that many Buddhists are also vegetarians. So these two words are semantically related and it is reasonable to recommend a vegetarian group to a person interested in Buddhism. While CTR fails to do this, our CET model successfully recommends this group based on using semantic similarity between words.

We also show the top words of the topics learned by CET and CTR in Table 5.3. As we can see, topics learned by CET look much neater. We can find some noisy

CET	dance dancing salsa tango salsa-dancing latin-dance flamenco dance-lessons ballet latin-dancing
	hiking excursionismo-hiking kayaking camping outdoors snowshoeing skiing backpacking walkers paddling
	dogs puppy cats pets chihuahua pug yorkie sheltie dachshund dog-lovers
	language culture spanish-culture english french-culture language-and-culture languages japanese-language german-culture european-culture
	movies films movie film movie-nights arthouse movies-dinner movies-and-dinner cinema-and-films dinner-and-a-movie
CTR	dance wellness group-fitness-training japanese dance-lessons cloud-computing english-conversation python democrat korean
	hiking outdoor-recreation startup-ventures javascript creative-writing new-york-city dogs singles-who-love-to-travel activities css
	business-networking weightloss stress foodie crosscultural socializing-dogs dog-lovers london liberty anime
	social language theater bike beer backpackers business-and-social-networking museum rockclimbing men
	fitness movies movie-nights exercise-nutrition business film snowboard cinema-and-films movies-dinner mountain-biking

Table 5.3: Top words of sampled topics learned from Meetup data by CET and CTR.

words in topics learned by CTR. For example, dance and Japanese are in the same topic and hiking and dogs are also in the same topic. Previous work has shown that LDA, which is used to model text in CTR, is not able to learn topics well when documents are very short [96]. The average number of tags for each meetup group is only 7, so it is really hard for LDA to learn good topics. However, by using the embedding vectors, which carry semantic meanings of words, CET can cluster word much better and learn neater and more meaningful topics.

5.3.2 Product Recommendation in Online Review Website

For the second experiment, we use data from Amazon, which is composed of 9 datasets used in [57]. We have users' explicit ratings at scale 1-5 of items and their

dataset	#users	#items	#reviews	#word types	#tokens/review
office	691	313	4034	12652	46.33
patio	748	344	6814	8691	32.7
software	314	235	2468	14317	83.03
beauty	4281	1817	33290	22208	33.91
sports	8039	5545	91294	37645	30.23
tools	4935	3346	38998	68390	55.14
toys	3479	2776	25951	51224	50.07
games	9919	6124	88684	301829	115.83
health	4529	2460	35123	39674	40.36

Table 5.4: Dataset statistics, which show number of users, number of items, number of reviews, total number of word types, average number of tokens per review in each column.

reviews. Similar to [57], we use the aggregated reviews of an item as the associated text of it. Users and items with fewer than 3 reviews are filtered out. Statistics of this type of dataset are shown in Table 5.4. We choose two state-of-the-art techniques that model both explicit ratings and text information as our baselines.

HFT: Hidden Factors as Hidden Topics [57] is a model that directly ties each dimension of hidden factors in matrix factorization of ratings to one hidden topic in review text by using an exponential transformation function.

RMR: Ratings Meet Reviews [49] is a model similar to HFT except the way they link ratings with reviews. It assumes that each user has one Gaussian rating distribution on each topic, which characterizes how the user is interested in this topic.

Quantitative Study: Rating Prediction

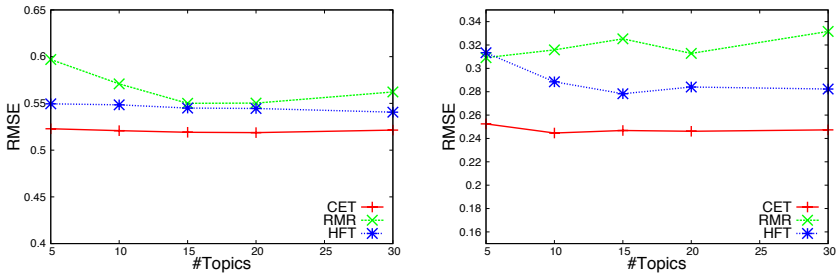
For Amazon review dataset, we use RMSE (Root Mean Squared Error) [57] as the evaluation metric, which is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{r}_i - r_i)^2}, \tag{5.18}$$

where r_i is the true rating for the ith testing instance and $\hat{r}_i$ is the prediction. The results over all 9 datasets are shown in Table 5.5. We can see that CET significantly

	office	patio	software	beauty	sports	tools	toys	video	health
CET	$\mathbf{0.521}^{\dagger}$	$\mathbf{0.252}^{\ddagger}$	$\mathbf{0.725}^{\ddagger}$	$\mathbf{0.371}^{\ddagger}$	$\mathbf{0.215}^{\ddagger}$	$\mathbf{0.746}^{\ddagger}$	$\mathbf{0.967}^{\ddagger}$	1.183	$\mathbf{0.483}^{\ddagger}$
RMR	0.597	0.309	0.767	0.484	0.351	0.802	1.013	**1.138**	0.595
HFT	0.552	0.283	0.776	0.444	0.262	0.813	1.146	1.172	0.548

Table 5.5: RMSE of CET, HFT and RMR. For each dataset, the best result is in bold font. † indicates that CET significantly outperforms RMR at 1% level. ‡ indicates that CET significantly outperforms both RMR and HFT at 1% level.



(a) (b)

Figure 5.2: RMSE over topic numbers on two datasets. The left one is *office* dataset, the right one is *patio* dataset.

outperforms RMR and HFT on most datasets. It means our model can effectively learn users' interests by modeling rating and text information.

To have a closer look at how the performance of all three models change over different number of topics, we pick two datasets *office* and *patio* and show the results in Figure 5.2. We can see that CET outperforms both baselines when using different numbers of topics. Its performance is also more stable over topic numbers compared with the other two.

	office	patio	software	beauty	sports	tools	toys	video	health
CET	-1.495	-1.198	-1.140	-1.730	-1.938	-1.776	-1.934	-1.434	-1.821
RMR	-0.744	-0.745	-0.627	-0.702	-0.711	-0.742	-1.247	-0.882	-0.793
HFT	-0.737	-0.739	-0.500	-0.718	-0.773	-0.957	-0.987	-0.752	-0.844

Table 5.6: Topic Coherence

Quantitative Study: Topic Coherence

We also evaluated our models with topic coherence, which is a metric measuring aspect quality based on co-occurrence of words [60]. It is defined as

$$C(t, V^{(t)}) = \frac{2}{M(M+1)} \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(v_l^{(t)})},$$

where $V^{(t)}$ contains the M most probable words in topic t. $v_m^{(t)}$ and $v_l^{(t)}$ are the mth and lth words in $V^{(t)}$. $D(v_l^{(t)})$ is the number of documents containing word $v_l^{(t)}$ and $D(v_m^{(t)}, v_l^{(t)})$ is the number of documents containing both $v_m^{(t)}$ and $v_l^{(t)}$.

As topic coherence is calculated based on co-occurrence statistics of words in the training data, CET may not get a good performance on this metric. Table 5.6 shows the results of CET, RMR and HFT. We can see that the performance of CET is not as good as that of RMR and HFT. However, as CET leverage semantic meanings encoded by word vectors, it can still learn clear topics, which we will show later.

Qualitative Study

In this subsection, we show the top words of some sampled topics learned by CET, RMR and HFT in Table 5.7. All topics are from the *office* domain and the number of topics is set to 30 for all models. As we can see, CET can learn meaningful topics like office, file, paper, purchase and so on as well as HFT and RMR. By taking a closer look at the top words of these topics, we can find that the top words of CET are cleaner. Most of the top words are about the same topic and there is less noise in these words. However, there exist some noisy words in the top word list of HFT and RMR, many of them are general words like “one”, “use”, “well”, etc. By using word vector to represent words, words can be clustered better compared with models like HFT and RMR. It is interesting that CET also discover a topic, family members, which cannot be learned by RMR and HFT. This may be a topic worth mining for recommendation as it probably reflects who the product is bought for.

Topics learned by CET					
work	tape	paper	product	pages	daughter
office	file	binder	products	templates	old
job	files	printed	price	interface	home
desk	tapes	printing	buy	page	son
working	folder	printer	purchase	multiview	mother
phone	folders	print	buying	text	father
cabinet	video	binders	brand	functionality	niece
offices	taped	pencil	purchasing	webpage	grandmother
works	filing	sheets	brands	app	granddaughter
telephone	clips	ink	pricing	template	dad
Topics learned by HFT					
desk	folders	binder	pen	cards	black
keyboard	tabs	binders	markers	paper	color
mouse	folder	rings	fine	card	folders
pad	file	pages	pens	print	look
hp12c	reinforced	open	colors	business	good
feet	tab	one	ink	avery	colors
rest	manila	pockets	write	printer	great
wrist	use	ring	sharpie	printed	nice
holder	smead	front	use	quality	side
platform	box	plastic	highlighters	make	well
Topic learned by RMR					
desk	folders	binder	markers	cards	folders
keyboard	files	binders	colors	paper	black
mouse	hanging	rings	pens	card	color
pad	using	open	ink	avery	look
rest	still	pockets	pen	print	file
wrist	drawer	pages	sharpie	business	good
holder	pendaflex	ring	highlighters	printer	great
feet	bottom	front	great	printed	nice
platform	product	avery	write	make	one
tray	capacity	cover	marker	professional	colors

Table 5.7: Top words of sampled topics learned by CET, HFT and RMR.

However, CET is not perfect and it fails to discover the topic about pens.

5.4 Discussion

In this chapter, we have proposed a recommendation model for social media based on users' ratings, text and word embedding vectors. Compared with existing work, our model is able to find the similarity between two pieces of text based on their semantic similarity rather than simply lexical similarity. This makes it more effective for recommendation problems in social media. Extensive experiments on two recommendation problems in social media show that this model can outperform state-of-the-art methods. A closer look at topics also tells us that by using the semantic meanings reflected in embedding vectors, our model can learn cleaner topics. When documents have as few as seven words on average, our model can still learn meaningful topics and get good recommendation results.

We have shown that using vectors learned from neural network based model can improve both recommendation and topic discovery in social media. It would be interesting to try vectors learned by other word embedding models such as topical word embedding [53], multi-prototype word embedding [65] and so on. Besides, it is a promising direction to model text in other ways beyond bag of words. Models which take order into consideration, like Recursive Neural Network [84] and Convolutional Neural Networks [40] are worth trying.

Chapter 6

A Neural Network Model for Semi-supervised Review Aspect Identification

In Chapter 4 and Chapter 5, we introduce our work on aspect word identification and aspect word clustering respectively. Actually, these two steps can be naturally handled by one joint model as they are closely related. However, few work has looked into this direction. In this chapter, we propose an integration of Neural Topic Model (NTM) and Recurrent Neural Network (RNN) to jointly identify and cluster aspect words in product reviews. As we have discussed, aspect word clustering is usually solved in an unsupervised manner, and topic models have been widely used for the task. A key difference of our neural network model from topic models is that we do not use multinomial word distributions but instead embedding vectors to generate words. Furthermore, to leverage review sentences labeled with aspect words, a sequence labeler based on RNN is incorporated into our neural network. The resulting model can therefore learn better aspect representations. Experimental results on two datasets from different domains show that our proposed model can outperform a few baselines in terms of aspect quality, perplexity and sentence clustering results.

6.1 Introduction

Sentiment analysis of online customer reviews has been well studied for over a decade. One of the key tasks in mining customer reviews is aspect identification [50]. Here aspects refer to features, components and other criteria on which a product or service may be evaluated by online users. Since the seminal work in [27], aspect identification has been recognized as a central problem in mining and summarizing customer reviews. Given a collection of reviews from the same domain (e.g., reviews of restaurants), aspect identification aims to discover a set of aspects, each associated with a set of aspect terms (or a distribution over such terms). For example, from restaurant reviews, we may expect to discover an aspect on service, with aspect terms such as "waiter" and "serve," and another aspect on food, with aspect terms such as "pizza" and "burger." The aspect identification task is useful for downstream tasks such as aspect-based review summarization [104] and product comparison [55].

Aspect identification is generally treated as an unsupervised task and a commonly adopted solution is based on topic models such as LDA (Latent Dirichlet Allocation) [2]. Here each aspect is modeled as a topic, which is essentially a multinomial distribution over words, and reviews are modeled as mixtures of these topics. A number of special topic models have been proposed for aspect identification [25, 64, 103].

With recent advances in neural networks and representation learning for natural language processing, embedding words in a low-dimensional hidden space to capture their distributional behaviors has shown to be effective for a number of data mining tasks [15, 99, 102]. In this chapter, we explore how neural network models can be used to address the review aspect identification problem and whether they can outperform standard topic models. Our work is motivated by two observations: (1) Compared with the traditional multinomial word distribution based language models, neural language models constructed in a continuous space may better handle

low-frequency words in reviews and address the data sparsity problem. (2) Sometimes review sentences with aspect terms annotated are available. For example, the Aspect Based Sentiment Analysis task in SemEval-2014 provides such annotated data. It has been shown that neural network models can achieve strong results on the supervised aspect term extraction task [51, 101]. We would like to explore how these trained neural network models can be used to help the aspect identification task.

In this work, we propose a neural network model for review aspect identification. Different from existing topic model based approaches to aspect identification, our model is based on continuous space language models, and it uses a small amount of labeled review sentences to train an RNN model for semi-supervised learning. Using reviews from two different domains, we show that our model improves the quality of the identified aspects compared with some baseline models, and both components of our proposed model contribute to the improved performance.

6.2 Method

In this section, we present our neural network model for aspect identification.

6.2.1 Problem Formulation

The setup of our aspect identification task is as follows. We assume that we have a set of unlabeled reviews $\mathcal{R}$ from the same domain, e.g., a set of restaurant reviews. In addition, we have a set of review sentences $\mathcal{S}$ from the same domain annotated with aspect terms, as shown in Table 6.1. Our goal is to discover K aspects from $\mathcal{R}$ and $\mathcal{S}$, where each aspect is associated with some parameter $\mathbf{v}_k$ and from $\mathbf{v}_k$ we can understand the meaning of the k^{th} aspect. In traditional topic model-based approaches to aspect discovery, each $\mathbf{v}_k$ would be a distribution over the words in the vocabulary, and the words with the highest probabilities in $\mathbf{v}_k$ would well represent the aspect. In our work, we do not constrain $\mathbf{v}_k$ to be a probability distribution, as

From the [**appetizers**] we ate, the [**dim sum**] and other variety of [**food**], it was impossible to criticize. The [**design**] and [**atmosphere**] are just so good.

Table 6.1: Examples of annotated sentences. Aspect words are highlighted and enclosed with brackets.

we will explain below.

6.2.2 Model Overview

The general idea behind our model is as follows. We aim to re-construct the reviews in $\mathcal{R}$ from a set of parameters capturing various properties of the reviews. To re-construct a review, we treat the review as a bag of sentences and generate the sentences one by one in a probabilistic way. Each sentence will probabilistically be assigned an aspect, and then be treated as a bag of words sharing the same aspect.

Different from standard topic models, however, we also model the context of each word using a recurrent neural network (RNN) and the context will be used to influence the probability of generating the word. Specifically, the probability of generating a word comes from a combination of a number of vectors representing different aspect models and a background model. This kind of a mixture model is inspired by [103]. However, our model has notably the following differences from [103]: (1) Unlike [103], which is an extension of LDA, we do not use multinomial distributions to model topics (i.e., aspects in this case). Instead, we use a neural networks with continuous vectors to derive the probabilities of generating different words. This treatment is similar to a number of recent work on neural topic models [6, 66]. (2) Unlike [103], which uses a Maximum Entropy model to incorporate the context of word into its probabilistic modeling, we use an RNN to incorporate the context, which presumably is more effective given the recent success of using RNN models for sequence modeling problems.

6.2.3 Review Generation Process

Modeling Aspects

We assume that there are K underlying aspects. Similar to [103], which assumes that each aspect has two word distributions, namely an aspect word distribution and an opinion word distribution, we assume that each aspect k has two embedding vectors associated with it: $\mathbf{v}_k \in \mathbb{R}^d$ and $\mathbf{c}_k \in \mathbb{R}^d$. Here $\mathbf{v}_k$ is meant to capture words that directly describe the aspect (as those highlighted terms in Table 6.1), such as "pizza" and "cake" for the aspect on food or "waiter" and "waitress" for the aspect on service. $\mathbf{c}_k$ is meant to capture other words closely associated with the aspect but are not considered opinion target terms. These may include "delicious" and "tasty" for the aspect on food or "friendly" for the aspect on service. Note however that neither $\mathbf{v}_k$ nor $\mathbf{c}_k$ is a distribution over the words in the vocabulary, and we will explain later how they are used to generate words.

Modeling Background Words

We assume that there is a background distribution over words, which we denote with $\boldsymbol{\theta}^{\text{b}}$. This distribution represents how reviews may contain words not related to any aspect.

Modeling Documents

Similar to [103], we assume that each review has a multinomial distribution over the K aspects. Let us use $\boldsymbol{\beta}_r$ to represent this distribution for the r^{th} review. We also assume that there is a document-independent probability λ that controls how likely a word is associated with an aspect or with the background model $\boldsymbol{\theta}^{\text{b}}$.

Modeling Word Context

We use $w_{r,s,n}$ to represent the n^{th} word in the s^{th} sentence in the r^{th} review. Here $1 \leq w_{r,s,n} \leq V$ is an index in the vocabulary and V is the vocabulary size. We

assume that this word has a vector $\mathbf{h}_{r,s,n}$ that encodes its context using an RNN model we will describe later. With this vector $\mathbf{h}_{r,s,n}$ and the RNN model, there is a probability $\pi_{r,s.n}$ associated with word $w_{r,s,n}$ to indicate how likely this word is an opinion target term rather than an opinion term, i.e., how likely $w_{r,s,n}$ is going to be generated from some $\mathbf{v}_k$ or from some $\mathbf{c}_k$.

Review Generation

With the various embedding vectors and probabilities defined above, we now describe the re-construction loss function which we try to minimize in order to learn the parameters. We use the negative log likelihood of generating the words inside all the reviews in $\mathcal{R}$ as our objective function. The overall objective function is as follows:

$$
\begin{aligned}
-\log p(\mathcal{R}) &= -\sum_{r=1}^{|\mathcal{R}|} \log p(\mathbf{w}_r) = -\sum_{r=1}^{|\mathcal{R}|} \sum_{s=1}^{M_r} \log \sum_{k=1}^{K} \beta_{r,k} p(\mathbf{w}_{r,s}|k), \qquad (6.1) \\
p(\mathbf{w}_{r,s}|k) &= \prod_{n=1}^{N_{r,s}} p(w_{r,s,n}|k) \\
&= \prod_{n=1}^{N_{r,s}} \Big[(1-\lambda)\theta^{\mathrm{b}}_{w_{r,s,n}} + \\
&\quad \lambda\Big(\pi_{r,s,n}\phi_{k,w_{r,s,n}} + (1-\pi_{r,s,n})\psi_{k,w_{r,s,n}}\Big)\Big], \qquad (6.2)
\end{aligned}
$$

where M_r is the number of sentences in the r^{th} review, $N_{r,s}$ is the number of words in the s^{th} sentence in the r^{th} review, $\mathbf{w}_r$ represents all the words in the r^{th} review, $\mathbf{w}_{r.s}$ represents all the words in the s^{th} sentence in the r^{th} review, and $\boldsymbol{\phi}_k$ and $\boldsymbol{\psi}_k$ are two distributions corresponding to aspect terms and opinion terms, which we will explain below.

Basically the loss function above shows that to generate a review r, for each sentence in the review we pick an aspect k according to the distribution $\boldsymbol{\beta}_r$. Then for each word in this sentence, we generate it either from the background model $\boldsymbol{\theta}_{\mathrm{b}}$ or one of the two models $\boldsymbol{\phi}_k$ and $\boldsymbol{\psi}_k$.

So far the model above is very similar to [103]. However, $\boldsymbol{\phi}_k$ and $\boldsymbol{\psi}_k$ are modeled differently from [103]. Instead of treating these as multinomial distributions and directly learning the probabilities, we assume that they are derived from the embedding vectors $\mathbf{v}_k$ and $\mathbf{c}_k$ as follows:

$$\boldsymbol{\phi}_{kv} = \frac{\exp(\mathbf{v}_k^\intercal \cdot \mathbf{w}_v^A)}{\sum_{v'} \exp(\mathbf{v}_k^\intercal \cdot \mathbf{w}_{v'}^A)} \tag{6.3}$$

$$\boldsymbol{\psi}_{kv} = \frac{\exp(\mathbf{c}_k^\intercal \cdot \mathbf{w}_v^C)}{\sum_{v'} \exp(\mathbf{c}_k^\intercal \cdot \mathbf{w}_{v'}^C)}. \tag{6.4}$$

$\mathbf{W}_A \in \mathbb{R}^{d \times V}$ and $\mathbf{W}_C \in \mathbb{R}^{d \times V}$ are two matrices to model the semantic representations of words, which are initialized with pre-trained Google word2vec.[1] Each column in them is used to encode one word type.

6.2.4 RNN to Incorporate Context

We now explain how we obtain $\pi_{r,s,n}$ for each word $w_{r,s,n}$ by making use of the annotated review sentences. Our method is again inspired by the MaxEnt-LDA model [103], in which a Maximum Entropy model was trained on some labeled data to help separate aspect words, opinion words and background words. The same idea applies to our problem, but here we use a Recurrent Neural Network (RNN) model, which represents the state of the art for aspect term extraction [51].

The motivation of making use of the labeled review sentences is that there are some patterns we can learn to locate aspect terms. For example, nouns following adjectives which are sentiment words, such as the word "service" in the phrase "excellent service," are more likely to be aspect terms. We can try to learn such patterns from the labeled review sentences, even though the labels only indicate which words are aspect terms but do not group them into aspects.

Because usually there is only a small amount of such labeled review sentences, to address the data sparsity problem, here we again make use of dense vector representations to train a classifier. Specifically, we use Recurrent Neural Network

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

(RNN) models. Let us assume that $(\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_n)$ is the sequence of words in a labeled sentence, where each $\mathbf{l}_i \in \mathbb{R}^d$ is a dense word embedding vector. Let $(y_1, y_2, \ldots, y_n)$ represent the corresponding labels marking the positions of the aspect terms. We can build an RNN model from the sequence $(\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_n)$ as follows:

$$\mathbf{h}_i = f(\mathbf{U}\mathbf{h}_{i-1} + \mathbf{V}\mathbf{l}_i + \mathbf{e}), \tag{6.5}$$

where $f(\cdot)$ is a non-linear activation function, $\mathbf{U} \in \mathbb{R}^{d_o \times d_o}$, $\mathbf{V} \in \mathbb{R}^{d_o \times d}$ and $\mathbf{e} \in \mathbb{R}^{d_o}$ are parameters to be learned, d_o is the output dimension and $\mathbf{h}_i$ is the hidden state at position i. We can then use $\mathbf{h}_i$ to predict the label y_i through a softmax layer. While there exist some other RNN structures like LSTM(Long Short Term Memory), Bidirectional-RNN, Bidirectional-LSTM and so on, RNN has simpler structure and competitive performance [51]. So we only use RNN to predict $\pi_{r,s,n}$ in this work.

To train this model, we maximize the probabilities of the observed labels in the training dataset $\mathcal{S}$. Given a new sentence, we can use the trained RNN model to obtain the hidden states $\mathbf{h}$, and for each word in the sentence, we can use its corresponding hidden state to obtain a probability $\pi_{r,s,n}$ for the word to be an aspect term.

6.2.5 Connections with Topic Models

With certain configurations, our model is closely connected with traditional topic models. However, our model learns aspect vectors and uses a linear transformation followed by the softmax function to model topic-word dependencies. Compared with multinomial distributions, which are typically used in topic models, our model can incorporate more information, like semantic meanings of words and topics. In recent years, neural network based topic models have been invented to incorporate pre-trained word embeddings [6, 46, 66]. Compared with these models, our model

Algorithm 1 Gibbs-EM algorithm for learning

for $i \leftarrow 1, maxEpoch$ **do** ▷ $maxEpoch$ is the maximum number of epochs.
  E-step:
  for $r \leftarrow 1, |\mathcal{R}|$ **do**
    for $s \leftarrow 1, M_r$ **do**
      Sample an aspect $t^i_{r,s}$ according to Formula 6.6.
    end for
  end for
  M-step:
  Keep $\mathbf{T}_i$ fixed. Compute the gradient $\frac{\partial \mathcal{L}_i}{\partial \mathbf{\Theta}}$ by back-propagation.
  Use the gradient to update all parameters $\mathbf{\Theta}$.
end for

is a more general framework. Each component of it can be replaced with other suitable options. So it is easier to extend and adapt to different tasks. Besides this, we uses RNN to separate aspect words from context words, which can potentially help us learn better topics. This has not been used in existing neural topic models.

6.2.6 Learning

To learn our model, we need to find the optimal values of $\mathbf{v}_k$, $\mathbf{c}_k$, $\boldsymbol{\theta}^b$, $\boldsymbol{\beta}_r$, $\mathbf{W}_A$, $\mathbf{W}_c$ and λ that can minimize the objective function $\mathcal{L} = -\log p(\mathcal{R})$.

Back-propagations cannot be directly used to learn our neural network as there are some constraints placed on β_r. To deal with this, one alternative is variational-EM algorithm. However, it is not an exact estimation algorithm as it tries to optimize the lower bound of the objective function. Instead of using variational inference to approximate posterior distributions at the E-step, we adopt Gibbs sampling to sample an aspect for the s^{th} sentence in the r^{th} review according to

$$p(t_{r,s} = k) = \frac{\beta_{r,k} p(\mathbf{w}_{r,s}|k)}{\sum_{k'} \beta_{r,k'} p(\mathbf{w}_{r,s}|k')}. \tag{6.6}$$

Then, in the M-step, we first update $\boldsymbol{\theta}_b, \boldsymbol{\beta}_r$ and λ based on the sampling results of

E-step:

$$\lambda_0 = \frac{n_0 + \eta}{n_0 + n_1 + 2\eta} \tag{6.7}$$

$$\boldsymbol{\theta}_{bw} = \frac{n_{bw} + \xi}{n_{b\cdot} + V\xi} \tag{6.8}$$

$$\boldsymbol{\beta}_{rt} = \frac{n_{rt} + \alpha}{n_{r\cdot} + T\alpha} \tag{6.9}$$

where n_0 is the number of words assigned to background, n_1 is the number of words assigned to context, n_{bw} is the number of times word w is assigned to background and n_{rt} is the number of words in review r that are assigned to topic t. $n_{b\cdot} = \sum_w n_{bw}$ and $n_{r\cdot} = \sum_t n_{rt}$. η, ξ and α are all hyper-parameters. Then, we apply back-propagation to update all other parameters in our neural network with the sampled aspect for sentence fixed. The objective function for the M-step in the ith epoch is

$$\mathcal{L}_i = -\log p(\mathcal{R}|\mathbf{T}_i) = -\sum_{r=1}^{|\mathcal{R}|}\sum_{s=1}^{M_r} \log p(\mathbf{w}_{r,s}|t^i_{r,s}), \tag{6.10}$$

where $\mathbf{T}_i$ is the sampled aspects of all sentences in epoch i and $t^i_{r,s}$ is the sampled aspect in epoch i for the s^{th} sentence in the r^{th} review. The gradient of our parameters can be calculated as follows:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_t} = \sum_v n^a_{tv}(\boldsymbol{\phi}_{tv}\boldsymbol{w}^A_v - \sum_{v'} \boldsymbol{\phi}_{tv}\boldsymbol{\phi}_{tv'}\boldsymbol{w}^A_{v'}) \tag{6.11}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{c}_t} = \sum_v n^c_{tv}(\boldsymbol{\psi}_{tv}\boldsymbol{w}^C_v - \sum_{v'} \boldsymbol{\psi}_{tv}\boldsymbol{\psi}_{tv'}\boldsymbol{w}^C_{v'}) \tag{6.12}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^v_A} = \sum_t n^a_{tw}\phi_{tv}(1-\phi_{tv})\boldsymbol{v}_t \tag{6.13}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^v_C} = \sum_t n^c_{tw}\psi_{tv}(1-\psi_{tv})\boldsymbol{c}_t \tag{6.14}$$

where n^a_{tv} is the number of times word v is assigned to aspect topic t, n^c_{tv} is the number of times word v is assigned to context topic t.

6.3 Experiments

In this section, we evaluate our proposed model from different angles. Through the evaluation we mainly want to test if our neural network model using aspect and context vectors to generate words work better than traditional topic models based on multinomial unigram word distributions for aspect identification. In addition, we also look at the generative ability and the effectiveness of clustering sentences using our model.

We consider the following different models for comparison.

- **LDA**: Latent Dirichlet Allocation. This is a classical topic modeling technique proposed in [2]. To have a fair comparison with our models and the other baselines, we use a modified version that assumes that all words in the same sentence come from the same topic.
- **ME-LDA**: LDA with Maximum Entropy classifier [103]. This models uses both traditional topic models based on multinomial unigram word distributions and Maximum Entropy models for supervision.
- **RNN-LDA**: LDA with RNN. We replace the maximum entropy classifier in ME-LDA with the trained RNN model to estimate the probability of each word being an aspect word or not. By comparing with this model, we can evaluate the effect of using aspect and context vectors together with softmax to generate words.
- **ME-NA**: Neural network for aspect identification with Maximum Entropy. This is a variation of our model. We replace LDA in ME-LDA with our neural network model. By comparing with this model, we can evaluate the usefulness of using RNN instead of standard linear classifiers for the supervision.
- **RNN-NA**: Neural network for aspect identification with RNN. This is our complete model as presented in Section 6.2, where we use both unlabeled and labeled data for aspect identification. We do not fine tune $\mathbf{W}_A$ and $\mathbf{W}_C$,

i.e., the word embeddings are not updated during training.

- **RNN-NA-t**: This is also our complete model RNN-NA. However, we initialize $\mathbf{W}_A$ and $\mathbf{W}_C$ with word embeddings and fine-tune them during training.

To compare the models above, we first conduct three experiments to evaluate the quality of identified aspects. Then we do a quantitative evaluation based on perplexity to check the model's ability to predict words in unseen reviews. We also do another quantitative evaluation using sentence clustering to evaluate each model's effectiveness in grouping review sentences into different aspects.

6.3.1 Data

We use two datasets for our experiments. The first one contains restaurant reviews from the Yelp academic dataset.[2] As the original dataset contains millions of reviews from different businesses, we only keep the restaurant reviews and randomly sample 20,000 from them. The other dataset is a laptop dataset crawled from Amazon, used by [92].[3] For the set of labeled training sentences, we use the sentences tagged with aspect terms from SemEval competitions. For the restaurant domain, the training sentences are from SemEval 2014 and 2015, and for the laptop domain, the training sentences are from SemEval 2015.

To pre-process the review data, we remove stop words and words with no pre-trained embeddings. Sentences with less than 3 words are also removed. After preprocessing, the Yelp dataset contains 17948 reviews, with each document containing 9.1 sentences on average and each sentence containing 5.8 words on average. In the Laptop dataset, there are 31,363 documents, where each document has 8.8 sentences on average and each sentence has 7.6 words on average.

[2] https://www.yelp.com.sg/dataset_challenge
[3] http://www.cs.virginia.edu/~hw5x/dataset.html

Dataset	#Aspect	LDA	ME-LDA	RNN-LDA	ME-NA	RNN-NA	RNN-NA-t
Yelp	10	0.533	0.65	0.45	0.50	0.53	**0.65**
	20	0.55	0.51	0.40	0.50	**0.63**	0.55
Laptop	10	0.47	0.50	0.70	0.70	0.58	**0.73**
	20	0.57	0.59	0.74	0.65	0.55	**0.75**

Table 6.2: Model precision (*MP*) of word intrusion by various models.

6.3.2 Aspect Quality

Word Intrusion

To evaluate the quality of aspects identified by our models, we conduct the word intrusion experiment [7]. For each discovered aspect, we extract 5 most probable words. We also extract another intrusion word that has a high probability in some other aspect but low probability in the current aspect. There words are then mixed and presented to the annotators to pick out the intrusion word. We ask four graduate students for the annotation. Fleiss' Kappa, which is a standard way to measure agreement among more than two annotators, shows that the inter-annotator agreement is 0.353 for the Yelp dataset and 0.487 for the Laptop dataset. These two scores indicate fair agreement and moderate agreement respectively. Model Precision (MP) is used as the evaluation metric, which is defined as

$$MP = \frac{1}{N}\sum_{a=1}^{N}\frac{M_a}{T}. \tag{6.15}$$

Here, N is the number of annotators, T is the number of aspects, M_a is the number of intrusion words that are correctly identified by annotator a.

The performances of all models with aspect number set to be 10 and 20 are shown in Table 6.2. We can see that RNN-NA-t performs the best most of the time, which demonstrates that our model is effective in mining aspects with high quality. RNN-NA can only outperform RNN-NA-t in one case. It proves that fine-tuning word embeddings in our model is important.

Dataset	#Aspect	LDA	ME-LDA	RNN-LDA	RNN-NA	ME-NA	RNN-NA-t
Yelp	10	-2.932	-4.421	-4.110	-0.757	-0.639	**-0.363**
	20	-3.487	-4.319	-4.129	-0.698	-0.628	**-0.443**
Laptop	10	-1.638	-5.476	-5.591	-1.090	-1.077	**-0.866**
	20	-2.746	-5.514	-5.698	-1.186	-1.111	**-0.787**

Table 6.3: Topic coherence.

Coherence

Besides human evaluation, we also evaluated our models with topic coherence, which is a metric measuring aspect quality based on co-occurrence of words [60]. It is defined as

$$C(t, V^{(t)}) = \frac{2}{M(M+1)} \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(v_l^{(t)})}, \tag{6.16}$$

where $V^{(t)}$ contains the M most probable words in topic t. $v_m^{(t)}$ and $v_l^{(t)}$ are the mth and lth words in $V^{(t)}$. $D(v_l^{(t)})$ is the number of documents containing word $v_l^{(t)}$ and $D(v_m^{(t)}, v_l^{(t)})$ is the number of documents containing both $v_m^{(t)}$ and $v_l^{(t)}$.

Table 6.3 displays the averaged topic coherence of different models. All models based on our proposed neural network can get better performance than others. Meanwhile, RNN-NA-t consistently gets the best performance. It proves that aspects discovered by our models are more coherent than those discovered by the competitors.

Qualitative Evaluation

To qualitatively study the quality of aspects identified by our proposed model, we show 4 sample aspects of the laptop dataset identified by RNN-NA-t and ME-LDA in Table 6.4. The top 10 most probable words of each aspect are displayed. Words that are closely related to the aspect are emphasized in bold font. From the table we can see that aspects learned by RNN-NA-t look more coherent and more words are closely related to the topic. As the dataset we use focuses on the laptop domain,

RNN-NA-t				ME-LDA			
network	display	os	support	network	display	os	support
wifi	**screen**	**windows**	**support**	windows	**screen**	**windows**	**warranty**
wireless	**display**	**os**	**service**	screen	keyboard	**system**	**service**
connection	**resolution**	**system**	**customer**	support	windows	**os**	**customer**
internet	keyboard	**operating**	**warranty**	**wireless**	battery	screen	**support**
windows	**color**	software	**tech**	**wifi**	quality	**operating**	drive
driver	**size**	**xp**	**shipping**	**connection**	**display**	software	screen
card	quality	**vista**	samsung	system	sound	use	hard
network	**colors**	use	screen	**internet**	price	keyboard	windows
drivers	**brightness**	works	battery	battery	touch	drive	battery
support	**retina**	hardware	system	keyboard	drive	battery	**shipping**

Table 6.4: Sampled learned aspects from the Laptop dataset.

some frequent words become dominant and appear in many topics, such as "windows" and "keyboard". This increases the difficulty of learning clean aspects from this dataset. However, compared with ME-LDA, RNN-NA-t is much less affected by this. Dominant words only appear near the bottom of the topic word list learned by it. The qualitative evaluation shows the advantage of our neural network for aspect identification in discovering meaningful and coherent aspects.

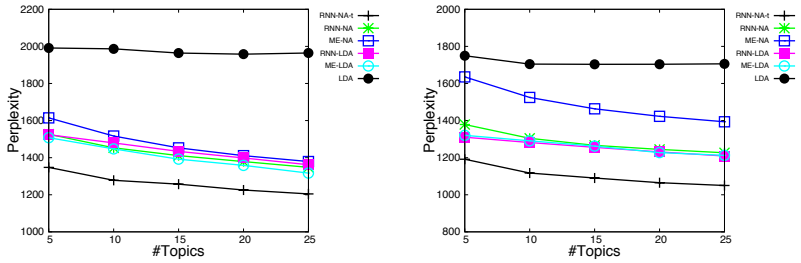
6.3.3 Perplexity

We evaluate all models' generative abilities using perplexity, which is a commonly used metric to evaluate the quality of language models and topic models. The definition of perplexity is as follows:

$$\text{perplexity} = \exp(-\frac{1}{N}\sum_{s\in\mathcal{T}} P(s)), \tag{6.17}$$

where $\mathcal{T}$ is our held-out test dataset, N is the total number of sentences in it and $P(s)$ is the probability of generating sentence s. In our experiment, we leave 20% of our dataset for testing and train the models based on the remaining 80% dataset. Perplexities over different numbers of aspects are shown in Figure 6.1.

We can see that our complete model with fine tuning of word embeddings is performing the best over various numbers of aspects on both datasets. Meanwhile, using RNN models to help separate aspect words from the rest performs better than using Maximum Entropy based models most of the time. Both findings verify that



(a) Yelp Restaurant (b) Laptop

Figure 6.1: Perplexities over different numbers of aspects for different models.

using neural networks in our model can improve generalization capabilities.

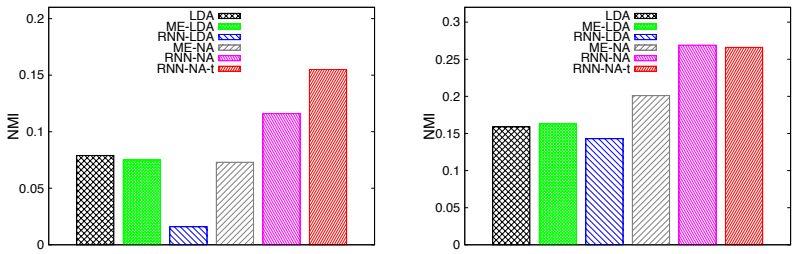
6.3.4 Sentence Clustering

To show how topical embeddings learned by different models benefit downstream tasks, we compare the different models in terms of sentence clustering. We manually labeled 100 sentences from the Yelp dataset and 100 sentences from the Laptop dataset. Normalized mutual information [56], which is a popular metric in text clustering, is used to measure performances in our experiment. we do not include it in this evaluation.

The results are shown in Figure 6.2. We can see that our proposed neural network models outperform all other competitors. As all sentences are from the same domain, it is uneasy to effectively discover clear aspects and cluster sentences by using co-occurrence statistics. So traditional topic models perform poorly. By learning topic embeddings, our models can improve a lot. Figure 6.2 also shows that using RNN to help separate out aspect words is much more effective than Maximum Entropy classifier.

6.4 Discussion

We explored aspect identification from reviews by proposing a novel neural network model. Our model is able to associate aspects and words using distributional vectors. An RNN model trained on labeled sentences is embedded into our model,



(a) Yelp (b) Laptop

Figure 6.2: Normalized mutual information.

which helped the model learn cleaner and more discriminative topics. Experiments on two datasets from different domains show that our model is effective in discovering meaningful aspects, predicting words and benefiting downstream applications such as sentence clustering. In the future, we will explore more complex neural network layers to model aspects and documents, and to jointly train the RNN with the neural network model for aspect identification.

Chapter 7

Dissertation Conclusion and Future Work

7.1 Summary

With the popularity of online shopping, there are more and more online reviews of products/services generated every day. As it is almost impossible to manually read these reviews, we need to develop automated techniques to process them. Aspect discovery is one fundamental way to help us do it. Aspects are attributes or features of a product/service, which people usually use to evaluate a product/service. They are also the dimensions along which people write reviews. Aspect discovery can benefit a lot of downstream applications such as aspect-based opinion mining, product recommendation, aspect specific rating prediction and so on.

We argue that there are 3 sub-tasks in aspect discovery from online reviews, which are: 1) define aspect, 2) identify aspect words, and 3) cluster aspect words. The first step aims at specifying the aspects we are interested in and figuring out how to find them. The second step is to find out words that are most relevant to aspect discovery and filter out other words. The last step is to cluster aspect words into groups. Words that refer to the same aspect should be grouped together in this step.

While many researchers have been looking into the second and third problem, few has studied the first one. Most previous work of aspect discovery from product reviews assumes that the topics learned by unsupervised topic models are naturally aspects. However, this may not be true as reviews also contain information other than aspects. To tackle the second problem, existing work usually transforms it into a sequence labeling problem and uses sequence labeling models to solve it. Most of these models are extensions of Conditional Random Filed or Hidden Markov Models. In recent years, Recurrent Neural Network (RNN) and its extensions have also been applied to this problem. Although these models can achieve good performances when the testing data come from the same domain as the training data, they cannot work well in a cross-domain setting. The main reason of this problem is that these models heavily rely on domain-dependent features. To solve the third problem, many variants of Latent Dirichlet Allocation have been designed. However, they treat words as discrete signals without considering their semantic meanings. Because of this, aspects can only be learned based on co-occurrence statistics of words. They may not work well when the word usage of reviews is diverse or the input data is sparse.

To solve the above problems of existing work, we use four chapters to introduce four models in this thesis. Firstly, we argue that besides aspects reviews also talk about properties of products, which should be separated from aspects. Properties are products' intrinsic features, such as their brands, genres and so on. We design a generative model which considers aspects, properties, and other related latent factors to model the generation of product reviews. Our model can separate aspects from properties and link both of them to ratings of reviews. Secondly, to improve the performance of RNN in cross-domain aspect word identification, we use domain-independent syntactic rules to generate auxiliary labels and design two neural network models to incorporate this information. With the help of auxiliary labels and our specially designed neural networks, we are able to learn features that are useful to both source and target domains. Thirdly, we design a topic model us-

ing representational vectors of words to learn aspects. Instead of treating words as independent discrete signals, our model can incorporate their semantic meanings. In the end, we also propose a model to jointly identify and cluster aspect words. In this model, RNN is seamlessly integrated with a neural network based topic model, which is also called Neural Topic Model (NTM).

7.2 Future Work

We acknowledge that this thesis does not exhaustively solve aspect discovery from product reviews. There are some open problems worth exploring in the future. We try to spot and introduce some of them in this section.

A Joint Training of Aspect Word Identification and Clustering

While aspect word identification and clustering are two tasks that are often separately handled by previous work, they are naturally related and may reinforce each other. We propose a model in Chapter 6 by integrating RNN with Neural Topic Model (NTM). The RNN model is first trained on a small scale labeled dataset and its output is then used as a distant supervision to guide the NTM we design. It is easy to imagine that words that can easily form an aspect are more likely to be aspect words. Meanwhile, words that are more likely to be aspect words should be able to form an aspect with higher probability. Our model only models the second hypothesis. We can model both hypothesises by jointly training the RNN and NTM components of our model at the same time. Both labeled sentences and unlabeled product reviews should be used in the joint training. Besides, an advanced optimization algorithm, variational back propagation is required to learn this model.

Modeling the Aspects and Words in a New Way

To model the semantic meanings of words, we develop two models in Chapter 5 and Chapter 6. We assume that each aspect has a representational vector to model its semantic meaning. Relatedness between a word and an aspect is determined by the inner product between the word vector and aspect vector. Compared with existing work, which uses multinomial distribution to represent an aspect, our models can use semantic level information. However, using inner product to model semantic relatedness is still too simple. A more advanced deep neural network should be used for it. By doing this, we may be able to model words and aspects in a higher dimension space without introducing too many parameters.

Modeling Word Sequence and Sentence Sequence in Product Reviews

In this thesis, we treat a document either as a bag of words or as a bag of vectors. These assumptions help us simplify the generation of product reviews without losing too much key information. However, the sequential ordering of words and sentences are also important. Even with the same words, the meaning of a sentence can be totally different when the order changes. Besides, there are transitions of aspects between words and sentences. Modeling the transitions can help us discover aspects more accurately. Existing work [34, 81] has tried to use directed probabilistic graphical (e.g. HMM) to do this and has achieved good results. It is a promising direction to use deep learning techniques to model word sequences, sentence sequences, and aspect transitions.

Bibliography

[1] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2–8, 2014.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine learning research*, 3:993–1022, 2003.

[3] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, 2007.

[4] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, 2006.

[5] S. Brody and N. Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812, 2010.

[6] Z. Cao, S. Li, Y. Liu, W. Li, and H. Ji. A novel neural topic model and its supervised extension. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2210–2216, 2015.

[7] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 22*, pages 288–296, 2009.

[8] G. Chen and L. Chen. Recommendation based on contextual opinions. In *User Modeling, Adaptation, and Personalization - 22nd International Conference*, pages 61–73, 2014.

[9] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, pages 767–774, 2012.

[10] Z. Chen, A. Mukherjee, and B. Liu. Aspect extraction with automated prior knowledge learning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 347–358, 2014.

[11] Z. Chen, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Discovering coherent topics using general knowledge. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, pages 209–218, 2013.

[12] Z. Chen, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Exploiting domain knowledge in aspect extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1655–1667, 2013.

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

[14] S. Clinchant and F. Perronnin. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 100–109, 2013.

[15] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[16] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee. How opinions are received by online communities: A case study on amazon.com helpfulness votes. In *Proceedings of the 18th International Conference on World Wide Web*, pages 141–150, 2009.

[17] R. Das, M. Zaheer, and C. Dyer. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics*, pages 795–804, 2015.

[18] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 193–202, 2014.

[19] Y. Ding and J. Jiang. A joint model of product properties, aspects and ratings for online reviews. In *Recent Advances in Natural Language Processing*, pages 131–137, 2015.

[20] Y. Ding and J. Jiang. Modeling social media content with word vectors for recommendation. In *Social Informatics - 7th International Conference*, pages 274–288, 2015.

[21] Y. Ding and J. Jiang. Towards opinion summarization from online forums. In *Proceedings of Recent Advances in Natural Language Processing*, 2015.

[22] Y. Ding, C. Yu, and J. Jiang. A neural network model for semi-supervised review aspect identification. In *Proceedings of the 21th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2017.

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

[24] J. L. Elmanan. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[25] G. Fei, Z. Chen, and B. Liu. Review topic discovery with phrases using the pólya urn model. In *Proceedings of 25th International Conference on Computational Linguistics*, pages 667–676, 2014.

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, pages 1735–1780, 1997.

[27] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, 2004.

[28] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[29] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 873–882, 2012.

[30] X. Huang and W. B. Croft. A unified relevance model for opinion retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 947–956, 2009.

[31] M. Iyyer, A. Guha, S. Chaturvedi, J. Boyd-Graber, and H. Daumé III. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proccedings of The 2016 Conference of the North American Association for Computational Linguistics*, pages 1534 – 1544, 2016.

[32] J. Jagarlamudi, H. Daumé, III, and R. Udupa. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 204–213, 2012.

[33] N. Jakob and I. Gurevych. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1035–1045, 2010.

[34] J. Jiang. Modeling syntactic structures of topics with a nested HMM-LDA. In *2009 Ninth IEEE International Conference on Data Mining*, pages 824–829, 2009.

[35] P. Jiang, C. Zhang, H. Fu, Z. Niu, and Q. Yang. An approach based on tree kernels for opinion mining of online product reviews. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 256–265, 2010.

[36] W. Jin, H. H. Ho, and R. K. Srihari. Opinionminer: A novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1195–1204, 2009.

[37] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230, 2008.

[38] Y. Jo and A. H. Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 815–824, 2011.

[39] M. I. Jordan. Serial order: A parallel, distributed processing approach. *Advances in Psychology*, 121:471495, 1997.

[40] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1749–1751, 2014.

[41] J. S. Kessler, M. Eckert, L. Clark, and N. Nicolov. The 2010 icwsm jdpa sentment corpus for the automotive domain. In *4th International AAAI Conference on Weblogs and Social Media Data Workshop Challenge*, 2010.

[42] M. Kgebck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality*, pages 31–39, 2014.

[43] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[44] F. Li, M. Huang, and X. Zhu. Sentiment analysis with global topics and local dependency. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1371–1376, 2010.

[45] J. Li, M. Ott, C. Cardie, and E. Hovy. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1566–1576, 2014.

[46] S. Li, J. Zhu, and C. Miao. A generative word embedding model and its low rank positive semidefinite solution. In *In Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics*, pages 666–675, 2016.

[47] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 939–948, 2010.

[48] C. Lin and Y. He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 375–384, 2009.

[49] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 105–112, 2014.

[50] B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[51] P. Liu, S. Joty, and H. Meng. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, 2015.

[52] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. Event-based social networks: Linking the online and offline social worlds. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1032–1040, 2012.

[53] Y. Liu, Z. Liu, T. Chua, and M. Sun. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424, 2015.

[54] Y. Lu, P. Tsaparas, A. Ntoulas, and L. Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International Conference on World Wide Web*, pages 691–700, 2010.

[55] Y. Lu, C. Zhai, and N. Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th International Conference on World Wide Web*, pages 131–140, 2009.

[56] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[57] J. J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 165–172, 2013.

[58] P. Melville, W. Gryc, and R. D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284, 2009.

[59] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[60] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum. Optimizing semantic coherence in topic models. In *EMNLP*, pages 262–272, 2011.

[61] S. Moghaddam and M. Ester. ILDA: Interdependent lda model for learning latent aspects and their ratings from online product reviews. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 665–674, 2011.

[62] S. Moghaddam and M. Ester. On the design of lda models for aspect-based opinion mining. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 2012.

[63] S. Moghaddam and M. Ester. The flda model for aspect-based opinion mining: Addressing the cold start problem. In *Proceedings of the 22Nd International Conference on World Wide Web*, 2013.

[64] A. Mukherjee and B. Liu. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 339–348, 2012.

[65] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1059–1069, 2014.

[66] D. Q. Nguyen, R. Billingsley, L. Du, and M. Johnson. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313, 2015.

[67] H. Nishikawa, T. Hasegawa, Y. Matsuo, and G. Kikui. Opinion summarization with integer linear programming formulation for sentence extraction and ordering. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 910–918, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[68] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[69] M. P. O'Mahony and B. Smyth. Learning to recommend helpful hotel reviews. In *Proceedings of the Third ACM Conference on Recommender Systems*, pages 305–308, 2009.

[70] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 502–511, 2008.

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

[72] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86, 2002.

[73] J. Pennington, R. Socher, and C. Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[74] J. A. Perez-Ortiz and M. L. Forcada. Part-of-speech tagging with recurrent neural networks. In *Proceedings of International Joint Conference on Neural Networks*, volume 3, pages 1588–1592. IEEE, 2001.

[75] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 486–495, 2015.

[76] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 27–35, 2014.

[77] G. Qiu, B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37:9–27, 2011.

[78] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in Speech Recognition*, pages 267–296. 1990.

[79] J. Reisinger and R. J. Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117, 2010.

[80] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, pages 1257–1264, 2007.

[81] C. Sauper, A. Haghighi, and R. Barzilay. Incorporating content structure into text analysis applications. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 377–387, 2010.

[82] C. Sauper, A. Haghighi, and R. Barzilay. Content models with attitude. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 350–358, 2011.

[83] K. Schouten and F. Frasincar. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, pages 813–830, 2016.

[84] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[85] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, 2015.

[86] I. Titov and R. T. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 308–316, 2008.

[87] I. Titov and R. T. McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, pages 111–120, 2008.

[88] C. Toprak, N. Jakob, and I. Gurevych. Sentence and expression level annotation of opinions in user-generated discourse. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 575–584, 2010.

[89] H. M. Wallach. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 977–984, 2006.

[90] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448–456, 2011.

[91] F. Wang, W. Pan, and L. Chen. Recommendation for new users with partial preferences by integrating product reviews with static specifications. In *User Modeling, Adaptation, and Personalization - 21th International Conference*, pages 281–288, 2013.

[92] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 618–626, 2011.

[93] W. Wang, S. J. Pan, D. Dahlmeir, and X. Xiao. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[94] T.-L. Wong, L. Bing, and W. Lam. Normalizing web product attributes and discovering domain ontology with minimal effort. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 805–814, 2011.

[95] Y. Wu and M. Ester. FLAME: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 199–208, 2015.

[96] X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 1445–1456, 2013.

[97] Y. Yang and J. Eisenstein. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 672–682, 2015.

[98] Y. Yin, F. Wei, L. Dong, K. Xu, M. Zhang, and M. Zhou. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of the 25 International Joint Conference on Artificial Intelligence*, pages 2979–2985, 2016.

[99] S. Zhai, K.-h. Chang, R. Zhang, and Z. M. Zhang. Deepintent: Learning attentions for online advertising with recurrent neural networks. In *KDD*, pages 1295–1304, 2016.

[100] M. Zhang and X. Ye. A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 411–418, 2008.

[101] M. Zhang, Y. Zhang, and D. Vo. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 612–621, 2015.

[102] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang. Retweet prediction with attention-based deep neural network. In *CIKM*, pages 75–84, 2016.

[103] W. X. Zhao, J. Jiang, H. Yan, and X. Li. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65, 2010.

[104] J. Zhu, H. Wang, M. Zhu, B. K. Tsou, and M. Ma. Aspect-based opinion polling from customer reviews. *IEEE Transactions on Affective Computing*, 2(1):37–49, 2011.

[105] L. Zhuang, F. Jing, and X.-Y. Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 43–50, 2006.