Product Defect Discovery and Summarization from Online User Reviews

Xuan Zhang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Weiguo Fan, Chair
Edward A. Fox, Co-Chair
Bert Huang
Alla Rozovskaya
Alan G. Wang
Zhongju Zhang

August 31, 2018
Blacksburg, Virginia 24061

Keywords: Opinion Mining, Product Defect Discovery, Opinion Summarization, Topic Model, Deep Learning

Product Defect Discovery and Summarization from Online User Reviews

Xuan Zhang

ACADEMIC ABSTRACT

Product defects concern various groups of people, such as customers, manufacturers, and government officials. Thus, defect-related knowledge and information are essential. In keeping with the growth of social media, online forums, and Internet commerce, people post a vast amount of feedback on products, which forms a good source for the automatic acquisition of knowledge about defects. However, considering the vast volume of online reviews, how to automatically identify critical product defects and summarize the related information from the huge number of user reviews is challenging, even when we target only the negative reviews. As a kind of opinion mining research, existing defect discovery methods mainly focus on how to classify the type of product issues, which is not enough for users. People expect to see defect information in multiple facets, such as product model, component, and symptom, which are necessary to understand the defects and quantify their influence. In addition, people are eager to seek problem resolutions once they spot defects. These challenges cannot be solved by existing aspect-oriented opinion mining models, which seldom consider the defect entities mentioned above. Furthermore, users also want to better capture the semantics of review text, and to summarize product defects more accurately in the form of natural language sentences. However, existing text summarization models including neural networks can hardly generalize to user review summarization due to the shortage of tagged training data and architecture weakness.

In this research, we explore topic models and neural network models for product defect discovery and summarization from user reviews. Firstly, a generative Probabilistic Defect Model (PDM) is proposed, which models the generation process of user reviews from key defect entities including product Model, Component, Symptom, and Incident Date. Using the topics produced by PDM in these aspects, people can discover defects which are represented by those entities. Secondly, we devise a Product Defect Latent Dirichlet Allocation (PDLDA) model, which describes how negative reviews are generated from defect elements like Component, Symptom, and Resolution. The interdependency between these entities is also modeled by PDLDA. PDLDA answers not only what the defects look like, but also how to address them using the crowd wisdom hidden in user reviews. Finally, the problem of how to summarize user reviews more accurately, and better capture the semantics in them, is studied using deep neural networks, especially Hierarchical Encoder-Decoder Models.

For each of the research topics, comprehensive evaluations are conducted to justify the effectiveness and accuracy of the proposed models, on heterogeneous datasets. Further, on the theoretical side, this research contributes to the research stream on product defect discovery, opinion mining, probabilistic graphical models, and deep neural network models. Regarding impact, these techniques will benefit related users such as customers, manufacturers, and government officials.

Product Defect Discovery and Summarization from Online User Reviews

Xuan Zhang

GENERAL AUDIENCE ABSTRACT

Product defects concern various groups of people, such as customers, manufacturers, and government officials. Thus, defect-related knowledge and information are essential. In keeping with the growth of social media, online forums, and Internet commerce, people post a vast amount of feedback on products, which forms a good source for the automatic acquisition of knowledge about defects. However, considering the vast volume of online reviews, how to automatically identify critical product defects and summarize the related information from the huge number of user reviews is challenging, even when we target only the negative reviews. People expect to see defect information in multiple facets, such as product model, component, and symptom, which are necessary to understand the defects and quantify their influence. In addition, people are eager to seek problem resolutions once they spot defects. Furthermore, users also want to better summarize product defects more accurately in the form of natural language sentences. These requirements cannot be satisfied by existing methods, which seldom consider the defect entities mentioned above, or hardly generalize to user review summarization.

In this research, we develop novel Machine Learning (ML) algorithms for product defect discovery and summarization. Firstly, we study how to identify product defects and their related attributes, such as Product Model, Component, Symptom, and Incident Date. Secondly, we devise a novel algorithm, which can discover product defects and the related Component, Symptom, and Resolution, from online user reviews. This method tells not only what the defects look like, but also how to address them using the crowd wisdom hidden in user reviews. Finally, we address the problem of how to summarize user reviews in the form of natural language sentences using a paraphrase-style method.

On the theoretical side, this research contributes to multiple research areas in Natural Language Processing (NLP), Information Retrieval (IR), and Machine Learning. Regarding impact, these techniques will benefit related users such as customers, manufacturers, and government officials.

Acknowledgments

First and foremost, I would like to express my sincere appreciation to my advisors, Dr. Weiguo Fan and Dr. Edward A. Fox, two important people in my life. It is their continuous support, guidance, motivation, encouragement, patience, and kindness, that helped me successfully go through my Ph.D. program. I cherish every moment we spent together on research discussion, paper and dissertation writing, and project collaboration.

Besides my advisors, I would like to show deep gratitude to other professors in my committee: Dr. Bert Huang, Dr. Alla Rozovskaya, Dr. Alan G. Wang, and Dr. Zhongju Zhang. Special thanks go to Dr. Chandan Reddy, Dr. Wenqi Shen, Dr. Alan Abrahams, and Dr. Naren Ramakrishnan. It's fortunate to have a chance to collaborate with these professors, who gave me insightful suggestions and invaluable guidance in my research and courses.

I give my thanks to my fellow colleagues in the Digital Library Research Laboratory (DLRL) and Center for Business Intelligence and Analytics (CBIA): Zhilei Qiao, Liuqing Li, Yufeng Ma, Prashant Chandrasekar, Mi Zhou, Sunshin Lee, Mohamed Magdy, Xinyue Wang, Saurabh Chakravarty, Abigail Bartolome, Qianzhou Du, Ziqian Song, Yu Wang, Siyu Mi, Xiaomo Liu, Jian Jiao, etc. Also I thank other friends who helped me during my Ph.D. study: Ke Zhai, Liang Shan, Brandon Fan, Shuo Niu, Da Zhang, Shuaicheng Zhang, etc.

I give special thanks to Mr. Jason Cline and Dr. Sarah Lukens for the intern opportunity and their help during my internship in GE Digital, which was valuable industry research experience in a US company.

I must express heartfelt gratitude to my family - my wife, Juan Li, my parents, my daughter Xiran Zhang, and my brothers. It's not easy to come back to school and gain the doctor degree after spending many years in industry. Without their continuous support, patience, and sacrifice, I could not complete the long journey of Ph.D. pursuit.

Thanks go to the Department of Computer Science, Advanced Research Computing (ARC), and Pamplin College of Business for the funding and facilities which supported me through my Ph.D. research.

Thanks go to NSF for support through grant IIS - 1319578. Thanks also go to NCI Inc. and Mayfair Group for their support through grants.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 3

1.3 Hypothesis 3

1.4 Research Questions 4

1.5 Dissertation Organization 4

2 Literature Review **6**

2.1 Product Defect Discovery 6

2.2 Opinion Mining 7

2.3 Text Summarization 9

3 Probabilistic Defect Model **10**

3.1 Approach 11

3.1.1 Concepts 11

3.1.2 Data Preprocessing 12

3.1.3 Model Design 13

3.1.4 Model Inference 14

3.1.5 Inference Acceleration 17

3.2 Data 17

3.2.1 NHTSA Datasets for Qualitative Evaluation 18

3.2.2 NHTSA Datasets for Quantitative Evaluation 19

3.2.3 NHTSA Datasets for Defect-Recall Consistency Analysis 20
3.2.4 Amazon Datasets for Qualitative Evaluation 20
3.3 Evaluation 21
3.3.1 Qualitative Evaluation on NHTSA Datasets 21
3.3.2 Product Defect Discovery Performance Evaluation on NHTSA Datasets 25
3.3.3 Consistency Analysis on Product Defects and Mitigating Measures on NHTSA Datasets 27
3.3.4 Predictive Analysis on NHTSA Datasets 28
3.3.5 Qualitative Evaluation on Amazon Datasets 29

4 Multiple-Facet Latent Dirichlet Allocation for Product Defect Discovery 36
4.1 Approach 38
4.1.1 Entity Extraction 38
4.1.2 PDLDA Model 39
4.1.3 Model Inference 43
4.1.4 Implementation and Parameter Setting 46
4.2 Data 46
4.2.1 Preprocessing 47
4.2.2 Evaluation Criteria 48
4.2.3 Baseline Methods 49
4.3 Evaluation Results 51
4.3.1 Performance of Review Clustering 51
4.3.2 Performance of Topic Coherence 53
4.3.3 Qualitative Evaluation 55
4.3.4 Visualization of Topics 59

5 Abstractive Summarization of Product Reviews with Hierarchical Encoder-Decoder Neural Network 62
5.1 Approach 63
5.1.1 Distant Supervision 65

5.1.2 Abstractive Summarization Models 67
5.2 Evaluation 71
5.2.1 Datasets 71
5.2.2 Baseline Methods 72
5.2.3 Evaluation Criteria 73
5.2.4 Evaluation Settings 74
5.2.5 Evaluation Results 76

6 Contributions and Future Work **83**
6.1 Contributions 83
6.2 Publications 84
6.3 Future Work 85

References **86**

List of Figures

3.1 Plate Notation of Probabilistic Defect Model 14

3.2 Vehicle Recall Prediction Performance of PDM and LDA, measured by Weighted F1 30

4.1 Examples of product defect reviews (**bold**: symptom, *italic*: resolution). . . 36

4.2 Plate Notation of PDLDA Model 42

4.3 Extract Component Entities by Frequent Item Set Mining 48

4.4 Plate Notation of ILDA (Moghaddam and Ester, 2011) 49

4.5 Plate Notation of MaxEnt LDA (Zhao et al., 2010) 50

4.6 PMI coherence of symptom and resolution topics for FORD Focus. 54

4.7 PMI coherence of symptom and resolution topics for APPLE MacBook. . . . 54

4.8 PMI coherence of symptom and resolution topics for patient.info. 54

4.9 The most relevant reviews of defects. 58

4.10 Front Page of the Vehicle Defect Analysis Website. 59

4.11 Vehicle Model List. 60

4.12 Defects of a Vehicle Model. 60

4.13 Topics of a Defect of FORD Explorer 2002. 61

4.14 Representative Reviews of a Vehicle Defect. 61

5.1 A Distant Supervision Approach to Develop Multiple-Document Summarization Models 64

5.2 Distant Supervision Method 1 to Create Training Data 66

5.3 Distant Supervision Method 2 to Create Training Data 67

5.4 A Sequence-to-Sequence Neural Network with a RNN + FNN Encoder . . . 68
5.5 A Sequence-to-Sequence Neural Network with a RNN + CNN Encoder . . . 69
5.6 A Sequence-to-Sequence Neural Network with a RNN + CNN + Attention Encoder . . . 70
5.7 Bahdanau Attention Model for Machine Translation . . . 74
5.8 Opinion & Argument Abstract Generation Model, adapted from Wang and Ling (2016) . . . 75

List of Tables

3.1 NHTSA Review Datasets for Qualitative Evaluation 19
3.2 Datasets for Clustering Performance Evaluation 20
3.3 Statistics of Two Product Datasets from Amazon Reviews 21
3.4 Top Defects for Honda, Toyota, and Chevrolet Models 23
3.5 Clustering Performance of PDM, LDA, and K-Means 27
3.6 Consistency between Identified Defects and TSB/Recall Records 28
3.7 Top Defects of Google Chromecast Identified by PDM at Review Level . . . 31
3.8 Top Defects of Google Chromecast Identified by PDM at Sentence Level . . 32
3.9 Top Defects of Amazon Kindle Tablets Identified by PDM at Review Level . 33
3.10 Top Defects of Amazon Kindle Tablets Identified by PDM at Sentence Level 34
3.11 Relevance Rate of Topic Words/Phrases in Two Datasets 35

4.1 Definition of PDLDA Notations 40
4.2 Datasets for Qualitative and Topic Coherence Evaluation 47
4.3 Datasets for Clustering Performance Evaluation 47
4.4 Review clustering Precision, Recall, and F-1 (%) on FORD Focus Dataset. . 52
4.5 Review clustering Precision, Recall, and F-1 (%) on Apple MacBook Dataset. 52
4.6 Review clustering Precision, Recall, and F-1 (%) on patient.info Dataset. . . 52
4.7 Example of topics extracted by LDA, ILDA, MaxEnt, and PDLDA. (irrelevant words are underlined; "transm*" in row 4 stands for "transmission") 57

5.1 An Overview of AMDS Models with RNN-based Encoder 68
5.2 An Overview of AMDS Datasets 73

5.3 Hyper Parameters of Models 1-3 75
5.4 ROUGE-1 Metrics on Dataset 2 77
5.5 ROUGE-2 Metrics on Dataset 2 77
5.6 Summary Statistics on Dataset 2 78
5.7 ROUGE-1 Metrics on Dataset 1 78
5.8 ROUGE-2 Metrics on Dataset 1 79
5.9 Summary Statistics on Dataset 1 79
5.10 An Example of ⟨Review Cluster, Summary⟩Pair 80
5.11 Sample Summaries Generated by Different Models 81
5.12 Time Used for 50 Epochs Model Training on Dataset 2 82

Chapter 1

Introduction

1.1 Motivation

Discovering product quality issues has been a substantial topic of analytical research and empirical studies (Abrahams et al., 2012, 2015; August and Niculescu, 2013). Existing studies show that the traditional methods of quality issue discovery – such as laboratory and field experiments by firms – have been challenged (Buell et al., 2016). The recalls of Samsungs Galaxy Note 7 smartphone (since 2016) and Takata airbags (since 2013) are examples of voluntary quality disclosure based on consumer feedback. Recalls of Note 7[1] due to battery explosion risk could cost Samsung $9.5 billion sales loss, estimated by Nomura analysts[2]. As another case revealed by customers, the fatal defect of the Takata airbag[3], which led to 11 deaths and approximately 180 injuries before a recall initiated, affected 19 vehicle manufacturers and 42 million vehicles. Those recalls were initiated after the quality issues were publicly disclosed by users, rather than by the manufacturers themselves.

With the fast development of online communities and other types of social media, people can freely post their feedback on defects of products (Abrahams et al., 2013; Chen and Xie, 2008; Yan et al., 2015). This feedback is useful to other consumers for decision-making, to firm executives for improving their products, and to governments to enhance product quality administration. For example, Chen and Xie (2008) propose that user reviews work as a form of "sales assistance" to customers because they can use reviews to know the real quality of products, and reduce the uncertainty of risks and transaction costs. Abrahams et al. (2012) demonstrate that vehicle defect discovery from social media could improve automotive quality management. As user generated content (UGC) in social media surges explosively and spreads virally at an unprecedented speed, many companies seek to transform

[1] https://en.wikipedia.org/wiki/Samsung_Galaxy_Note_7

[2] http://money.cnn.com/2016/10/11/technology/samsung-galaxy-note-7-what-next/

[3] http://www.consumerreports.org/cro/news/2016/05/everything-you-need-to-know-about-the-takata-air-bag-recall/index.htm

business opportunities by uncovering hidden value from the content (Fan and Gordon, 2014; Luo et al., 2013; Yu et al., 2013).

However, identifying and summarizing critical product defects from tremendous volume of unstructured UGC is a nontrivial task, which was explored in only a very small number of prior studies. Researchers have struggled with the product defect identification problem, since it is not tractable for human text perusal, even for moderately sized textual content. For example, Abrahams et al. (2012) manually tagged 900 discussion threads from 1,316,881 in the HondaTech.com discussion forum. This sample is far less than 0.1% of the total number of discussion threads. Major studies still employ labor-intensive supervised methods to examine the product quality issues (Abrahams et al., 2012, 2015; Fan and Gordon, 2014). Yet, these classification-based methods can only predict the type of product issue in a given user review; it is hard to capture detailed information about defects as intensively reported in a review collection. In addition, the trained classification models are hard to generalize to different product domains. In one of the rare research studies which utilized unsupervised learning, Liang et al. (2017) applied topic modeling and association rule mining to get more information on product problems. Nevertheless, the proposed method focused on defect symptoms only, while ignoring other key aspects such as product model, defective component, and resolution.

Compared to product defect identification, which discovers the defect types or keywords from user reviews, the user review summarization task provides even richer information to users by generating summaries in the form of natural language sentences. Therefore, users can learn more information from the generated summaries, which have better readability. In this research, we concentrate on abstractive summarization from a collection of user reviews. The methods in the text summarization research area can be categorized into *extractive* and *abstractive* summarization based on the strategy. While extractive summarization relies on extraction of text snippets (e.g., key sentences, clauses, or phrases), abstractive summarization aims to produce a summary using paraphrase strategy. Thus, the words in abstractive summaries may be beyond the original reviews (Das and Martins, 2007). This research thus addresses an Abstractive Multiple-Document Summarization (AMDS) problem. The existing work (Liu et al., 2018; Wang and Ling, 2016) in this area is very limited. Since their models are trained using web pages and movie reviews, respectively, they are hard to generalize to the summarization of general product reviews.

To fill the above gaps in terms of product defect discovery, we propose novel topic modeling and deep neural network methods, which aim at capturing important defect entities, identifying representative reviews for each defect, and summarizing user reviews we are interested in. The basic idea of the topic models is: each topic model will generate a set of joint topics for each defect complained about in the review collection. For example, the Product Defect Latent Dirichlet Allocation (PDLDA) model in Chapter 4 will produce a "Component" topic, one or more "Symptom" topics, and one or more "Resolution" topics, for each defect. Together, these topics will help people understand what the defect looks like and how people tried to address it. The representative reviews retrieved for each defect will offer additional

information. The Probabilistic Defect Model has similar functions but distinctive features and structure. In contrast, the proposed deep neural network models take a user review collection as input, and use various Encoder-Decoder networks to produce an abstractive summary in the form of natural language sentences. Evaluations are conducted to prove the effectiveness and accuracy of these models in identifying product defects.

1.2 Problem Statement

The primary problem this research tries to address is how to recognize and summarize the most complained about product defects from massive sets of online user reviews. Since the target is to extract useful knowledge from customer comments, it can be considered as a kind of opinion mining research. This problem can be further decomposed into the following sub-problems.

First, we try to discover what the product defects look like from a given user review collection, including the key attributes (such as product model, component, and symptom) of product defects. To understand a commonly criticized product issue, we may want to know what kind of product model is involved. The problem is usually caused by one or more flawed components, which may reveal what has caused the issue. In addition, the defect symptoms reported by previous users may show us more details about the problem.

Then, we move one step further to identify and summarize how people resolve product issues by linking resolutions to issues. Besides the relevant components and symptoms, many people also mention or suggest solutions to the problem during product problem discussion. The resolution information is extremely useful to other users suffering from the same issue, and benefits manufacturers and mechanics who also confront the defects.

Finally, we summarize user reviews in which people have interest, in the form of natural language sentences using deep Neural Networks, particularly, Hierarchical Encoder-Decoder networks (Sutskever et al., 2014). Few papers (Liu et al., 2018; Wang and Ling, 2016) have investigated how to use deep learning to summarize a collection of documents, such as web pages or user reviews. An issue with these two methods is that both of them concatenate all the source documents from a collection into a long sequence, which is then taken as the input of their neural networks. This incorporates unnecessary and invalid dependency and order between the source documents, and may lead to loss of information when modeling very long sequences. Thus it may have a negative influence on summary generation.

1.3 Hypothesis

The main hypothesis of this research is that the three proposed methods can capture more information on product defects than existing methods, with better performance.

Compared with existing product defect discovery methods based on supervised learning, the proposed Probabilistic Defect Model (PDM) can capture more detailed information on the most complained about defects in a given review dataset. For each defect, the related product model, defective component, and symptom keywords can be extracted correctly by PDM. This model will achieve better performance than baseline unsupervised algorithms such as standard LDA and K-Means, when evaluated using clustering performance criteria such as Precision, Recall, and F1.

In contrast to existing studies, the proposed PDLDA model can identify defect resolutions from user reviews, in addition to the corresponding flawed components and problem symptoms. The interdependency among component, symptom, and resolution can be modeled accurately. The topic quality of PDLDA outperforms baseline methods such as standard LDA, ILDA (Moghaddam and Ester, 2011), and MaxEnt LDA (Zhao et al., 2010) when evaluated by topic coherence and clustering performance criteria.

The proposed hierarchical Encoder-Decoder neural network models are trained on datasets created for user review summarization. Using novel hierarchical encoder architectures (e.g., RNN+FNN or RNN+CNN), they are supposed to eliminate the improper dependencies between the source documents which are assumed by existing models (Wang and Ling, 2016; Liu et al., 2018). Thus, they should produce abstractive summaries with higher quality.

1.4 Research Questions

The overall research question for this study is: How and to what extent can we identify and summarize product defects from user review datasets with less human intervention? That question can be further decomposed into the following specific research questions:

- How to identify the most complained about product defects from user reviews, together with key attributes such as product model, component, and symptom, using joint topics?
- How to recognize critical defects revealed by a set of user reviews and find corresponding solutions suggested for them using interdependent topics?
- How to produce abstractive summaries of higher quality given a collection of user reviews?

1.5 Dissertation Organization

We research novel topic modeling methods and neural network models in solving the product defect discovery and summarization problem. The proposed models are developed based

on Probabilistic Latent Semantic Analysis, Latent Dirichlet Allocation, and Sequence-to-Sequence Neural Network, etc.

The remaining part of this dissertation is organized as follows. Chapter 2 reviews the literature in terms of product defect discovery, opinion mining, and text summarization. Chapter 3 introduces the Probabilistic Defect Model, which aims to capture key entities, such as product model, component, and symptom, of each defect. Chapter 4 shows how a novel LDA model (PDLDA) identifies and connects the component, symptom, and resolution of defects. Chapter 5 explores how Neural Network-based summarization models, especially Hierarchical Encoder-Decoder models, implement and improve the summarization of user review collections. Finally, Chapter 6 summarizes the contributions of this research, and discusses ideas for future work.

Chapter 2

Literature Review

First, the most relevant research area of this study is product defect discovery from user reviews: How to identify and describe the most complained about defects from a given collection of user comments. In addition, this research is also related to opinion mining. Product defect identification can be regarded as a special case of aspect-based opinion mining, since the objective is to extract useful knowledge (defect information) in multiple facets from user reviews. Finally, this research has connection with the text summarization research field as well, since we aim to produce abstractive summaries for a given review collection.

2.1 Product Defect Discovery

The problem of product defect discovery from customer reviews has been investigated by only a very small number of researchers. Few publications discuss the key facets of product defects, which are essential elements in product defect description and management.

The early research in this area was conducted in the automotive domain, then generalized to other product domains. Abrahams et al. (2012) proposed a set of "smoke words" to distinguish performance defects, safety defects, and non-defects. In a later study, they found that key terms, product features, and semantic features are useful factors for recognizing product defects, but stylistic, social, and sentiment features have little correlation with product defects (Abrahams et al., 2015). Product defect discovery research was extended to home appliances by Law et al. (2017). Instead of distinguishing performance defects, safety defects, and non-defects, another study by Abrahams et al. (2013) investigated how to recognize flawed components, one of the essential attributes of a product defect. However, a set of predefined categories is a strong prerequisite for all these supervised learning methods, no matter the defect types or component types. Thus, limited generalization and flexibility are their drawbacks. In contrast, Tutubalina and Ivanov (2014) introduced a syntactic approach to extract problem phrases from user reviews. They recognized problem indicators by search-

ing for negated verb phrases or problem words defined in lexicons, then extracted targets modified by these indicators according to syntactic dependencies among words. However, this method relies on pre-defined syntax dependency rules, which are difficult to generalize. A recent study (Liang et al., 2017) used topic modeling and association rule mining to get more information on product problems. However, the proposed method focused on defect symptoms, while ignoring other key aspects such as product models and defective components. Li et al. (2005) proposed an unsupervised probabilistic model to model the key entities of events. But the model and related entity extraction methods are designed for event recognition, which cannot be used for product defect discovery.

In the research of Abrahams et al. (2013) and Liang et al. (2017), key elements of product defects such as defective components and problem symptoms were studied. The related components usually explain the reason and the cause of the defect. For example, if smart phone customers always complain of a battery drain issue, the manufacturer should consider enhancing the battery in their next generation of product. On the other hand, the symptom text describes detailed information on how the issue is manifested, what kind of damage is caused, and how serious the issue is. One example is the "Phone signal is weak" symptom of Xiaomi smart phones mentioned in the paper of Liang et al. (2017). In addition, associating various defects with related product models is also important, especially when the complaints on different product models are mixed in one collection. Those are the essential aspects for product defect description and discovery. Nevertheless, no existing research is able to discover product defects from these comprehensive aspects.

2.2 Opinion Mining

As mentioned above, product defect mining is a special case of aspect-based opinion mining. Abundant research has been done in this area, in contrast to the situation regarding product defect discovery.

The earliest research in aspect-based opinion mining focused on opinion entity extraction, which extracts key entities (e.g., product feature, component, review sentiment, opinion granularity, etc.) from unstructured customer comments. The extracted entities can be used for summarization or other kinds of analysis later. Hu and Liu (2004) first identified product features using frequent item set mining, then took adjectives co-occurring with feature words as opinion words. A rule and lexicon based method for finding product properties and user opinions was presented by Popescu and Etzioni (2007). A clustering-based review summarization method was proposed by Ly et al. (2011). Supervised learning methods were reported in Jin et al. (2009); Li et al. (2010); Yang and Cardie (2013). These methods can identify aspect and opinion words based on manually labeled data. Nevertheless, the frequent item set mining method may face accuracy problems while the rule-based method and supervised learning methods rely on extensive manual work.

Afterwards, more researchers switched to LDA models for aspect-based opinion mining, where they were able to incorporate various factors of opinion, due to the flexibility of LDA. Titov and McDonald (2008) separated global and local opinion topics using a multiple-grain LDA model. Paul and Girju (2010) developed an LDA model which jointly discovers topics and aspects. Lin and He (2009) pointed out the importance of sentiment in terms of opinion mining and added it as a dimension of topic modeling. The ILDA model proposed by Moghaddam and Ester (2011) models product aspects and corresponding customer ratings. Lim and Buntine (2014) extends their work for Twitter opinion analysis. They replaced Dirichlet distributions with the Griffiths-Engen-McCloskey (GEM) (Pitman, 1996) distribution and Pitman-Yor process (PYP) (Teh, 2006); they also incorporated a sentiment prior from the lexicon. Zhao et al. (2010) proposed a Maximum Entropy-LDA hybrid model to separate aspect words and aspect-specific opinion words. Li et al. (2015) tried to break the limit of the "Aspect + Opinion" pattern by using verb expressions to capture opinion details. Recently a holistic model which included all of the above key entities (e.g., aspect, opinion, sentiment, and granularity) has been proposed by Wang et al. (2016). They further boosted their LDA model with domain knowledge learned by lifelong machine learning (Chen and Liu, 2014).

Depending on the main purpose of opinion mining LDA models, they can be separated into three categories: opinion summarization, rating prediction, and information retrieval. The first type of LDA models, such as (Moghaddam and Ester, 2011; Zhan and Li, 2011; Wang et al., 2016), target generating joint topics to describe and summarize user opinion. Thus, Park et al. (2015b) combined user reviews with product specifications using an LDA model to produce an augmented specification. A second group of works (Lin and He, 2009; Wang et al., 2010; Xue et al., 2015) focuses on the prediction of aspect ratings or sentiment classes. The last category of related work is information retrieval for user reviews. For example, Park et al. (2015a) proposed an LDA model which connected Mobile Application (APP) descriptions with corresponding reviews, in order to improve mobile APP searching. They modeled the dependence between two variables by taking the description's distribution as the prior of reviews.

Although many LDA models have been proposed for aspect opinion mining, the entities extracted and modeled by existing methods are insufficient for product defect mining. In particular, their design is for general opinion mining purposes, but it is hard to generalize to product defect entities, such as product model, component, symptom, and resolution. Also, their models could not adapt well to the characteristics of defect description. Their entity extraction methods can't identify the above mentioned defect entities correctly, because the locations of defect entities in user complaints vary from general opinion entities (e.g., aspect, sentiment, and granularity).

2.3 Text Summarization

Text summarization is condensing a source text into a short summary while preserving the essential information and general meaning of the original text. Text summarization can be divided into Single Document Summarization (SDS) and Multiple Document Summarization (MDS) according to the source text type (single vs. multiple document summatization), and categorized into extractive or abstractive summarization according to summary type. Extractive methods work by selecting the most representative snippets (e.g., sentences, clauses, phrases, etc.) in the original text to form the summary[1]. In contrast, abstractive methods first create a semantic representation of the source text, and then use natural language generation approaches to produce a summary in a paraphrase manner. Thus, the generated summary might involve verbal innovations and words unseen in the original text.

In pioneering research, Luhn (1958) introduced a method to extract salient sentences from the text using features such as word and phrase frequency. Graph theoretic representation is another popular approach Kruengkrai and Jaruskulchai (2003); Xu et al. (2013). When generating a summary, some hot sub-graphs with important nodes and more edges will be selected. Topic modeling techniques have been extensively used for MDS. For example, Daumé III and Marcu (2006) proposed BayeSum, a Bayesian model designed for query-oriented summarization. Wang et al. (2009) introduced a Bayesian sentence-level topic model which used not only term-document but also term-sentence correlations for summarization purpose.

Recently, deep learning methods with an RNN-based Sequence-to-Sequence model (See et al. (2017); Nallapati et al. (2016); Rush et al. (2015); Chopra et al. (2016); Hu et al. (2015)) have demonstrated impressive performance in terms of SDS. Deep reinforcement learning models (Chen and Bansal, 2018; Paulus et al., 2017) also demonstrate great potential in SDS. However, only a few papers (Liu et al. (2018); Wang and Ling (2016)) have investigated how deep learning performs on MDS tasks. An issue with these methods is that both of them concatenate all the source documents from a collection into a long sequence, which is then taken as the input of their neural networks. This incorporates unnecessary and invalid dependency and order between the source documents. Further, these models may suffer from information loss if they model very long sequences using RNN, which may have a negative influence on the summary generation. To tackle these problems, we propose several methods which use hierarchical encoders (e.g., RNN+FNN or RNN+CNN) to remove the improper dependencies and merge the semantic content of multiple documents in the given collection.

[1] `https://en.wikipedia.org/wiki/Automatic_summarization`

Chapter 3

Probabilistic Defect Model

The research question for this chapter is: how to identify the most complained about product defects from user reviews, especially negative reviews, together with key attributes such as product model, component, and symptom. Product defects may indicate potential product safety hazards and thus call attention from various interest groups such as customers, manufacturers, and regulators. For instance, in the recall event of Samsung Galaxy Note 7 smartphones[1] in 2016, the existing battery testing procedure failed to detect battery malfunction which led to battery explosion. This recall resulted in not only sales suspension but also significant financial loss ($9.5 billion sales loss estimated by Nomura analysts) to the company and its suppliers[2]. Recent studies discovered that user complaints (Abrahams et al., 2012, 2015), visible online as heavily negative postings, are indicative of product defects. Thus, manufacturers particularly have great interest in methodologies that can prevent and detect product defects in order to proactively address product quality issues, handle product recalls, and defend against liabilities. The recent surge of customer complaints in user-generated contents enables manufacturers to discover product defects at an early stage. However, existing methods mainly focus on classifying product defects into different categories, which does not provide comprehensive information for manufacturers to pinpoint potential defects and evaluate possible damage.

To address this problem, we propose a novel unsupervised product defect detection method, namely Probabilistic Defect Model (PDM). We get some inspiration from Li et al. (2005) during this research, but have changed the model design, entity extraction approach, and evaluation methods according to our research problem. Different from existing methods, PDM automatically extracts and summarizes comprehensive defect information from consumer complaints such as product model, affected component, and symptom.

This research makes the following contributions to the product defect discovery literature. First, we contribute methodologically by proposing a novel probabilistic defect model (PDM)

[1] http://www.businessinsider.com/samsung-issues-galaxy-note-7-battery-report-2017-1

[2] http://money.cnn.com/2016/10/11/technology/samsung-galaxy-note-7-what-next/

for extracting potential product defects with multiple-facet information from unstructured user complaints. Unlike prior works that merely screen the defect discussions from massive online reviews, we go one step further to pinpoint the detailed defect information, i.e., defect components, symptoms, and possible resolutions, if any. To the best of our knowledge, this is the first attempt to utilize a joint probabilistic model in the field of product defect discovery without much human intervention. Firms can greatly benefit from product defect discovery, including enhancing product quality with low cost and timely access to demand-side knowledge. Second, we evaluate the proposed model across different product domains to show the generalizability of our methodology. In addition to qualitative and quantitative evaluation on vehicle complaints, we conduct additional case studies using negative reviews on media players and tablet computers. Our results indicate that the suggested approach can be generalized to various product domains. Moreover, the predictive evaluation indicates the output of PDM has a better predictive power for defect mitigation measures (i.e., product recalls and Technical Service Bulletins). Third, our results are more directive and interpretable than traditional product defect methods. In addition to unigram (single-word) defect topics, we extract topics in phrases, which greatly improves the interpretability of the results. Non-technical business executives can easily develop product enhancement strategies based on our results. Lastly, our model takes into account the fact that one user complaint may discuss multiple defects, while one sentence usually focuses on one problem. Therefore, we run PDM at the sentence level rather than at the review level, which adds complexity to the analysis process but greatly improves the coherence and accuracy of the extracted topics in the reviews.

In this chapter, Section 3.1 introduces the proposed model for product defect identification, including the model design, the inference process of the model, and how we accelerate our model inference. Section 3.2 presents the datasets, including those used for the quantitative and qualitative evaluation. Finally, Section 3.3 demonstrates the results of the quantitative and qualitative evaluation in two case studies, and includes discussion on these results.

3.1 Approach

3.1.1 Concepts

Similar to the assumption of Latent Dirichlet Allocation (LDA), which presumes all documents in a corpus are generated from a set of "topics" (Blei et al., 2003), we assume that all defect complaints in a collection are generated based on a distribution of "defects". Therefore, a generative probabilistic model is developed to model this generative process in which "complaints" are observations and "defects" are latent variables. Unlike LDA which treats all of the words as of the same type, in PDM a user complaint is assumed to comprise four entities: {*Model*, *Component*, *Symptom*, *Incident Date*}. Each "defect" has its own word or time distribution in terms of each entity. For example, one defect may be highly related

with "Honda Accord" in terms of Model, closely related with "brake" and "pad" regarding Component, and have strong connection with "noise" and "fail" in terms of Symptom. Furthermore, this defect was frequently reported between 2008 and 2012 regarding Incident Date. These entities of complaints (observations) can be extracted from various product review datasets such as NHTSA (National Highway Traffic Safety Administration) databases, and negative Amazon reviews rated with one or two stars.

3.1.2 Data Preprocessing

The entities of reviews (observations) can be extracted from various product review datasets (e.g., NHTSA databases and Amazon reviews). Pre-processing technologies such as frequent item set mining, Part-Of-Speech (POS) tagging, regular expressions, or product component isolation (Abrahams et al., 2013; Fan and Gordon, 2014; Hu and Liu, 2004; Zhao et al., 2010) are applied for entity extraction. Usually, the product model and incident date can be extracted from corresponding metadata of the reviews. For example, product model information exists in the "MODELTXT" column of the NHTSA dataset, and also exists in the product ID attribute in the Amazon review dataset. The incident date can be easily found in the "DATEA" column of the NHTSA dataset, and the "reviewTime" attribute of the Amazon review records. The Component and Symptom entities can be extracted using the methods introduced in Section 4.2.1.

Since topic models (including PDM) are quite sensitive to noise (Cohen et al., 2014), we apply Term Frequency-Inverse Document Frequency (TF-IDF) and POS tagging to remove noise from the symptom description text before running PDM. Specifically, we use the TF and IDF in all the documents to calculate a value for each word. Here, TF is the total number of occurrences of the word in the collection. This is a little different from the traditional TF value, which is calculated within each document. We do this because we want to know the importance of each word in the entire corpus, not only in each document. In practice, this method works fairly well. Only nouns, verbs, and adjectives with high TF-IDF value are kept; these are assumed to be the most important words describing a defect symptom. This noise-removing step is needed for all the datasets, as noise exists everywhere.

Symptom word extraction can be completed at both unigram level and phrase level. The bigram symptom phrases are generally more informative than the unigram symptom words. We conduct an additional phrase extraction pre-processing step using the dependency parser of Stanford NLP (Chen and Manning, 2014). Most of the dependency patterns introduced in Moghaddam and Ester (2012) – including "AMO", "NSUBJ", "DOBJ", "ADVCL", "ADVMOD", "AUXPASS", "COP", "NN", and "NEG" – have been used. We add "ADVCL", "ADVMOD", and "AUXPASS" because verbs and adjectives are important in describing defect symptoms.

3.1.3 Model Design

As the primary assumption of PDM, the review records in a given dataset are generated from a set of "defects". Thus, a generative probabilistic model is developed to model their relationship, where "reviews" are observations and "defects" are latent variables. A review is assumed to comprise four entities: Model, Component, Symptom, Date; therefore, a defect has 4 corresponding distributions to generate those entities.

Multinomial and Gaussian distributions are chosen for modeling the defect entities above. The first three entities of a review are composed of words, which can be modeled as vectors with their own term space. For instance, the symptom word vector of a review x_i looks like an N-dimensional vector *[symptom word 1, ..., symptom word N]*, where each component s_c in this vector is the occurrence count of symptom word c in review x_i. If the number of terms in the symptom vocabulary is N, then N will be the length of the vector. In Natural Language Processing and Information Retrieval, words in a corpus are usually modeled using Multinomial distributions (Heinrich, 2008; Blei et al., 2003; Hofmann, 1999). Therefore, three multinomial distributions are designed for the three textual entities (i.e., Model, Component, and Symptom) of each defect. In contrast to the discrete distributions used to model textual entities, the Date entity can be modeled by a Gaussian distribution over time, as the reports on a defect may cover a long period. Considering the fact that multiple defects may be complained about at the same moment, a Gaussian Mixture Model (GMM) is chosen to model the Incident Date entity of all defects (Fisichella et al., 2010). Assuming the four entities of a review are independent, we calculate the joint probability of a review as:

$$P(review) = P(model) \cdot P(component) \cdot P(symptom) \cdot P(date) \tag{3.1}$$

We further assume the generation of a review record follows the process below.

Algorithm 1: Review generation process of PDM

Step1: Sample a defect d_j from $Mul(\theta)$.
Step2: Generate a review x_i with likelihood $P(x_i|d_j)$
for *each word W of* x_i **do**
  if *W is a Model word* **then**
    sample a *Model* word: $Model_{im}$ from $Mul(\varphi_M^j)$
  if *W is a Component word* **then**
    sample a *Component* word: $Component_{ic}$ from $Mul(\varphi_C^j)$
  if *W is a Symptom word* **then**
    sample a *Symptom* word: $Symptom_{is}$ from $Mul(\varphi_S^j)$
  Sample a Date: $Date_i$ from $N(\mu^j, \sigma^j)$
end

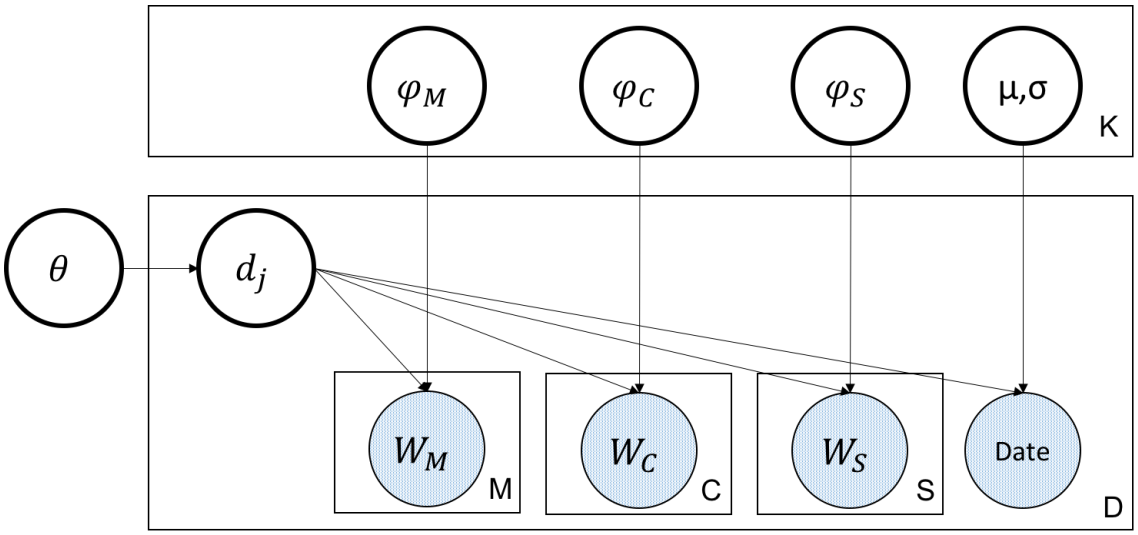


Figure 3.1: Plate Notation of Probabilistic Defect Model

The jth defect is represented by d_j in this algorithm, while the ith review is marked as x_i; the vector θ decides the probabilities of defects, which forms defect priors; φ_M^j, φ_C^j, and φ_S^j are parameters of multinomial distributions of defect d_j; μ^j and σ^j are parameters of the conditional Gaussian distribution given defect d_j.

Figure 3.1 shows the graphical diagram for this generative model, where W_M, W_C, and W_S mean the M Model words, the C Component words, and the S Symptom words in a review. They form the observation together with the date entity in this model. D is the number of reviews, while K is the number of defects.

3.1.4 Model Inference

Using Maximum Likelihood Estimation (MLE), the log-likelihood of the joint distribution of reviews can be written as:

$$Log(P(X|\theta) = Log(\prod_{i=1}^{D} P(x_i|\theta)) = \sum_{i=1}^{D} Log(\sum_{j=1}^{K} P(d_j)P(x_i|d_j,\theta_j)) \tag{3.2}$$

Here X stands for the collection of review records; D and K are the number of reviews and the number of defects, respectively.

These parameters can be estimated by the Expectation-Maximization (EM) algorithm (Dempster et al., 1977), which requires multiple iterations to search for the optimal value of those parameters. In practice, we set the EM converging condition as the log likelihood (Equation

(3.2)) difference between two iterations being no more than 1.0. According to our experiments, the EM algorithm stopping at this point produced good topics, which are further evaluated in Section 3.3. The EM algorithm consists of two steps, E-step and M-step. Usually, the E-step is used to calculate the posterior of the defects, while the M-step estimates and updates the parameters of related distributions.

In the **E-step**, we calculate the posteriors $P(d_j|x_i)$ using Equations (3.3), (3.4), (3.5), and (3.6):

$$P(d_j|x_i)^{(t+1)} = \frac{P(d_j)^{(t)}P(x_i|d_j)^{(t)}}{P(x_i)^{(t)}} = \frac{P(d_j)^{(t)}P(x_i|d_j)^{(t)}}{\sum_{k=1}^{K} P(d_k)^{(t)}P(x_i|d_k)^{(t)}} \tag{3.3}$$

Here (t) means the tth iteration. We need $P(x_i|d_j)$ to calculate Equation (3.3). According to Equation (3.1), we have:

$$P(x_i|d_j) = P(Model_i|d_j) \cdot P(Component_i|d_j) \cdot P(Symptom_i|d_j) \cdot P(Date_i|d_j) \tag{3.4}$$

The parts in Equation (3.4) can further be computed using methods (i) and (ii), given below.

(i) In the case of entities following a multinomial distribution (e.g., symptom):

$$P(Symptom_i|d_j) = \prod_{w}^{V} P(w|d_j)^{C_w} \tag{3.5}$$

V stands for the *symptom word* set in review i; each word $w \in V$ appears C_w times in review i. The probability $P(w|d_j)$ is initialized with a random value, such as $1/N$, where N is the size of the symptom words vocabulary.

(ii) In the case of a *date* entity which follows GMM:

$$P(date_i|d_j) = N(date_i|\mu_j, \sigma_j) \tag{3.6}$$

In Equation (3.6), μ_j is the mean of the jth Gaussian distribution, while σ_j stands for the standard deviation.

The **M-step** estimates and updates the parameters of the multinomial distributions and the GMM distribution, by maximizing the $Log(P(X|\theta))$ in Equation (3.2). Methods (i) and (ii) are used for different types of entities.

(i) In the case of entities following a multinomial distribution (i.e., Model, Component, Symptom):

$$P(w|d_j)^{(t+1)} = \frac{1 + \sum_{i=1}^{D}(P(d_j|x_i)^{(t+1)} \cdot tf(i,w))}{N + \sum_{i=1}^{D}(P(d_j|x_i)^{(t+1)} \cdot \sum_{s=1}^{N} tf(i,s))} \tag{3.7}$$

Here $tf(i,w)$ means the number of times word w occurs in review x_i, M represents the number of reviews, and N is the vocabulary size of that entity. We use Laplace smoothing (Heinrich, 2008) to avoid the zero probabilities for infrequent words.

(ii) In the case of Incident Date entity which follows GMM, we calculate the parameters using Equations (3.8) and (3.9):

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{D}(P(d_j|x_i)^{(t+1)} \cdot date_i)}{\sum_{i=1}^{D}(P(d_j|x_i)^{(t+1)})} \tag{3.8}$$

$$\sigma_j^{2^{(t+1)}} = \frac{\sum_{i=1}^{D}(P(d_j|x_i)^{(t+1)} \cdot (date_i - \mu_j^{(t+1)})^2)}{\sum_{i=1}^{D}(P(d_j|x_i)^{(t+1)})} \tag{3.9}$$

The defect proportions (priors) are updated as:

$$P(d_j)^{(t+1)} = \frac{\sum_{i=1}^{D} P(d_j|x_i)^{(t+1)}}{D} \tag{3.10}$$

Once the EM algorithm converges, we can use PDM to extract defect topics from user complaints or reviews. Moreover, each defect topic contains Model, Component, and Symptom words as well as a date. For example, we can obtain the most probable Symptom words of defect j using Equations (3.11) and (3.7). Similarly, we can also get the Component and Model entity words that are the most closely associated with this defect.

$$P(Symptom_w|d_j) \tag{3.11}$$

For example, the most probable symptom words of defect j can be calculated by Equations (3.11) and (3.7). We can also figure out the most probable words of Component and Model entities in a similar way.

$$y_i = \arg\max_j P(d_j|x_i) \tag{3.12}$$

After we have identified all defect topics, we can locate the most representative user complaints for each defect topic. Another way to understand a defect is to find the representative reviews closely related to it. Those representative complaints provide direct evidence for the manufacturer to understand the defect complaint in its original context. To do that, we first figure out the most relevant defect j given review i by Equation (3.12). Then, for each review x_i, a defect label y_i is assigned, which indicates its most related defect.

Finally, the reviews which have the highest $P(d_j|x_i)$ among all reviews assigned with the jth defect are good representatives of defect j.

Using the above methods, we are able to extract all the entities of a defect *(Model, Component, Symptom, Date)*, which constitute the key information of the most critical defects.

3.1.5 Inference Acceleration

In practice, this EM inference process is CPU-intensive. The time complexity of the inference algorithm is: $O(DefectCount \cdot reviewCount \cdot VocabularySize^2)$. Therefore, program acceleration is needed to achieve better performance, which is solved by the following methods.

The text pre-processing steps help accelerate the inference, besides reducing noise. The vocabulary size is reduced through removing useless words. Specifically, the Stanford NLP tagger (Toutanova et al., 2003) is used to do POS-tagging for a text; then only nouns, verbs, and adjectives are kept. Concurrent threads also have been used for acceleration.

3.2 Data

In this research, we gathered user reviews on vehicles, electronics, and tablet computers from two sources: NHTSA and Amazon.com. Derived datasets have been constructed for each dataset.

In the first study, we conduct a qualitative evaluation using NHTSA datasets to demonstrate the effectiveness of PDM in terms of identifying key product defects; then we conduct a document clustering experiment to compare the clustering accuracy of PDM and baseline methods (e.g., LDA and K-Means). In addition, we perform a predictive analysis in which the identified product defects produced by PDM and LDA are used to predict defect-mitigating measures, such as vehicle recalls and Technical Service Bulletin (TSB) records.

In the second case study, we extend the application of PDM to negative Amazon reviews on media players and tablet computers and show the generalizability of our method.

3.2.1 NHTSA Datasets for Qualitative Evaluation

In the experiments conducted in 2015, we used three databases (Complaint, Recall, and Technical Service Bulletin) of NHTSA for the primary evaluation.

The NHTSA Complaint database[3] had 1.13 million user complaints as of December 31, 2015, covering various vehicle brands and models sold in the United States. Each record has a number of attributes such as "Maker", "Model", "Year", "Component", "Description", "Fail date", "Injury count", "Death count", etc. The most common mitigating measures against vehicle quality problems include Technical Service Bulletin (TSB) and vehicle recall. The NHTSA recall database we used has 107K recall records, while there are 376K instances in the TSB database as of December 31, 2015. The recall database has special fields like Consequence description and Corrective action. A TSB is issued by a vehicle manufacturer to diagnose and repair vehicles when there are reports on an unexpected problem. The TSB database also has a special field which summarizes the discovered problem. An important difference between a TSB and a recall regarding vehicle products is that a recall is usually related with critical safety issues requested by organizations such as NHTSA. The recall and TSB databases are used as ground truth for the predictive evaluations, which are introduced in Section 3.2.3.

For the qualitative evaluation, we use complaint datasets on Honda, Toyota, and Chevrolet models (e.g., Accord, Civic, Pilot, etc.) produced between 2005 and 2009. There are several reasons to select these vehicle models. First, all of the three companies are major vehicle manufacturers (Abrahams et al., 2012). Second, these vehicle models have more complaints during this period than the other models.

Table 3.1 presents the summary of statistics of these three datasets. As shown in Table 3.1, the majority of the complaints happened between 2005 and 2009. Particularly, Chevrolet has the most vehicle models and complaints among the three manufacturers, while Honda has the smallest number of models and complaints. The complaints are expressed by short texts. The average sentence count in a complaint is between three and four, while the average number of words is between 46 and 49. The complaint length is similar for the three datasets, regardless of vehicle maker.

[3] `https://www-odi.nhtsa.dot.gov/downloads/flatfiles.cfm`

Table 3.1: NHTSA Review Datasets for Qualitative Evaluation

Dataset	Honda		Toyota		Chevrolet	
	2005-09	2010-15	2005-09	2010-15	2005-09	2010-15
Models	25	25	56	51	91	63
Reviews	13,327	3,572	29,241	7,752	34,191	7,650
Average sentences	3.38	3.27	3.35	3.24	3.35	3.26
Average tokens	46.78	48.39	47.34	48.21	46.68	48.00

3.2.2 NHTSA Datasets for Quantitative Evaluation

We use an indirect way to evaluate PDM accuracy in terms of product defect discovery, as a direct evaluation involves manually processing a large number of complaints and summarizing the multiple-aspect information of hidden defects, which can be inefficient and subjective. As a result, we adopt clustering accuracy as the primary measure in our experiments, as it is one of the most common performance evaluation criteria (Moghaddam and Ester, 2012) for unsupervised machine learning.

To evaluate the clustering accuracy of PDM and baseline methods, we need to manually create complaint clusters in which each cluster is formed by similar consumer complaints. However, since there are more than one million records in the NHTSA Complaint database, which makes it difficult to decide on the number of clusters and assign each complaint to one of them, we take vehicle recall records as the guidelines during the cluster construction process. After we label, prune, and synthesize the datasets, the size of these datasets is reduced to be more manageable than the datasets used in the qualitative evaluation. The process of synthesizing the evaluation datasets is as follows:

- Identify the top 10 critical recall records for each automobile manufacturer from the vehicle recall database. These records have the largest number of affected vehicles.
- For each critical recall record identified in the previous step, use its affected model and component as the SQL search conditions to retrieve corresponding complaints in the NHTSA complaint database.
- Combine the complaints related to the same recall record as a raw partition. We removed those complaints without any description or having multiple types of issues.
- Following the method of Zhao et al. (2010), we remove similar partitions (e.g., having similar components) to avoid controversy, and remove small partitions with less than

50 instances to avoid imbalanced data issue (since most partitions have more than 100 instances, a very small partition will cause an imbalanced data problem).

As the above manual synthesis work is needed, the size of these datasets is much smaller than those used in the qualitative evaluation.

Table 3.2 shows the datasets produced for evaluation.

Table 3.2: Datasets for Clustering Performance Evaluation

Vehicle Brand	**Number of reviews**	**Number of clusters**
Toyota	1514	6
Honda	1054	4
Chevrolet	1675	5

3.2.3 NHTSA Datasets for Defect-Recall Consistency Analysis

This experiment is treated as a binary classification problem, which predicts whether a complaint has a corresponding mitigating measure (i.e., Recall or TSB) or not. We use complaints on Honda models produced between 2005 and 2009 for the experiment, with datasets shown in Table 3.1. For each complaint, we find whether a corresponding Recall or TSB record exists, using an SQL query. If a recall or TSB record with the same model and component has been posted after that complaint was submitted, we assume the complaint is resolved by the corresponding mitigation measure(s), and mark it as a positive instance, otherwise negative. Only recalls or TSBs issued after the reported complaints are considered, since we try to predict a follow-up mitigating measure using existing complaints. Among 9740 complaints in the dataset, 977 have corresponding mitigation records.

3.2.4 Amazon Datasets for Qualitative Evaluation

To generalize our method to other product domains, we use negative reviews from Amazon review datasets[4] (McAuley et al., 2015a) in additional case studies. For illustration purposes, we use negative customer reviews from two types of products: media player and tablet computer. We only include reviews with one or two star ratings since we focus on product complaints. Amazon reviews with such low ratings are considered as negative reviews and are typically customer complaints on products (Yin et al., 2016). Therefore, we apply PDM to identify potential product defects embedded in these negative reviews. The electronics dataset has 2,554 complaints on the Google Chromecast HDMI Streaming Media Player, and the tablet dataset has 1,010 complaints for three Amazon Kindle tablets.

[4] `http://jmcauley.ucsd.edu/data/amazon/links.html`

Table 3.3 shows the statistics of the two products in Amazon datasets. Although the negative Amazon review datasets have fewer product models than are in the NHTSA dataset, these reviews have longer texts than those from NHTSA. The average number of sentences is 6.30 and 9.48 in the Google Chromecast dataset and Amazon Kindle dataset, respectively. Therefore, a negative Amazon review has a higher probability of discussing more than one type of defect, which makes it an interesting context to test the PDM model.

Table 3.3: Statistics of Two Product Datasets from Amazon Reviews

Dataset	**Google Chromecast 2013**	**Amazon Kindle 2012-13**
Models	1	3
Complaints	2,554	1,010
Average sentences	6.30	9.48
Average tokens	107.95	176.87

3.3 Evaluation

3.3.1 Qualitative Evaluation on NHTSA Datasets

We apply the PDM to discover defects from NHTSA complaints on Honda, Toyota, and Chevrolet models produced between 2005 and 2009. Most complaint entities such as Model, Component, and Date can be easily extracted from the corresponding fields in the Complaint database. Nevertheless, text pre-processing is necessary to extract the Symptom entity. In particular, we use TF-IDF and POS tagging to remove less important words from the symptom description of each complaint. Only nouns, verbs, and adjectives with high TF-IDF values are considered as the most essential words for defect discovery. Specifically, we select and use the top 2000 words after sorting about 5000 words by TF-IDF value in an original vocabulary dictionary. Using these configurations, we are able to extract reasonable topics with little noise, shown in Table 3.4, while keeping the model inference complete in acceptable time[5].

Next, we extract top defects, represented as a group of topic words, from each of the three datasets. In particular, ten defects, each represented as a group of topics, are extracted from the dataset of every manufacturer. The proposed method is able to identify the key information of each defect, including the most probable model, components, symptom words, and representative reviews. For each vehicle model i, we can also find its most related defects d_j by using Equations (3.13) and (3.14). Here, $P(Model_i|d_j)$ and $P(d_j)$ are figured out using Equations (3.5) and (3.10) during the PDM inference with EM.

[5]We run 100 EM iterations in 13 minutes on 40000 records with 50 Java threads using a 4-core i7 2.9GHz CPU.

$$d_j = \arg\max_j P(d_j|Model_i) \tag{3.13}$$

$$P(d_j|Model_i) = P(Model_i|d_j)P(d_j) \tag{3.14}$$

The symptom topic extraction has been done on both the unigram level and phrase level, as shown in Table 3.4.

As a demonstration of the outcome, we select the two most criticized vehicle models for each manufacturer based on the number of complaints received. Table 3.4 presents the top defect for each vehicle model along with the most likely related component, symptom keywords (unigrams and phrases), and top three representative complaints. For the 2006 Honda Civic, the top defect extracted from the user complaints is related to visibility (a component category defined by NHTSA). Based on the related symptom words, the visibility defect was caused by broken sun visors. The top defect of the 2008 Honda Accord is related to the service brake and the symptom words suggest a cause related to worn brakes. The top defect of the 2007 Toyota Camry is related to the accelerator while the 2007 Toyota Prius has a prominent issue related to its headlight failure. For Chevrolet models, a power steering issue stood out for the 2006 Cobalt while the 2005 Trailblazer suffered from a fuel leakage issue.

Compared to the unigram symptom topics, the phrase symptom topics have better readability. The additional phrase extraction pre-processing step is done using the dependency parser of Stanford NLP (Chen and Manning, 2014) with 10 dependency patterns: "ACOMP", "AMOD", "NSUBJ", "DOBJ", "ADVCL", "ADVMOD", "AUXPASS", "COP", "NN", and "NEG". Most of these dependency patterns are introduced in (Moghaddam and Ester, 2012). We add "ADVCL", "ADVMOD", and "AUXPASS" because verbs and adjectives are important in defect symptom description. Using the defects in Table 3.4, people can understand what kinds of problems are mostly complained about by customers, what kinds of components evidence a higher risk of failure in a vehicle model, and what types of symptoms they may produce. The defect entities (component, symptom words, and representative reviews) also reflect good consistency, thanks to the joint probability assumption of PDM.

Our results clearly show the effectiveness of the proposed method in the discovery of detailed product defect information from complaints. Compared to the generic topic modeling technique, PDM can produce rich information on the most complained about product defects of each vehicle model, such as the affected components, the related symptom phrases, and the representative complaints. The informative details of each discovered defect can help manufacturers quickly develop an investigation plan and a mitigation strategy once the problem is confirmed.

Table 3.4: Top Defects for Honda, Toyota, and Chevrolet Models

Maker	**Model**	**Defective Components**	**Symptom Topics (Unigrams)**	**Symptom Topics (Bigrams)**	**Description of Representative Reviews**
Honda	Civic 2006	Visibility	visor sun driver side window break stay replace windshield split	rear tire, civic hybrid, be problem, sun visor, visor split, be defective, visor fail, be issue, cause wear, control arm	1. The visor broke off with normal use.
					2. Passenger side sun visor came apart at the seam.
					3. My visors pop a part when using them.
	Accord 2008	Service brakes	brake rear tire wear replace pad noise problem need front	rear brake, make noise, wear Accord, when depressed, brake pad, brake fail, be soft, when kick, was depressed	1. 2008 Honda Accord rear brakes.
					2. Honda Accord 2008 brakes problem
					3. Rear brakes going out at 22,800 miles on 2008 Honda Accord.

Toyota	Camry 2007	Vehicle speed control	brake accelerate stop pedal speed acceleration accelerator foot engine parking	when accelerate, apply brake, bag deploy, when apply, brake fail, when depressed, suddenly accelerate, when stop, was involved, bag fail	1. Acc. Sticking, vehicle speed up.
					2. Car will suddenly speed up.
					3. Driving down highway when car accelerated on its own.
	Prius 2007	Exterior lighting	headlight turn light side back driver visor hide problem replace	turn back, when turn, headlight fail, headlight Prius, hid headlight, be problem, no reason, headlight turn, turn light	1. While driving at night both headlights went out.
					2. Headlights of my 2007 Toyota Prius would go off intermittently.
					3. My headlights went out twice this week on my 2007 Toyota Prius.

Chevrolet	Cobalt 2006	Steering	power steer steering turn cobalt wheel light fail problem failure	steering fail, make turn, when turn, light illuminate, turn wheel, was loss, power steering, steering failure, was unable, assist fail	1. I have a 2005 Cobalt SS with 62k on it and my power steering went out.
					2. Power steering went out.
					3. Power steering goes in and out.
	Trail-blazer 2005	Fuel system, gasoline	fuel gauge gas empty tank light fill full read work	fill tank, be full, tank full, when full, when fill, when start, be problem, gauge fail, was empty	1. Fuel gauge inoperable.
					2. Fuel gauge erratic/incorrect.
					3. Fuel gauge fluctuation.

3.3.2 Product Defect Discovery Performance Evaluation on NHTSA Datasets

In order to assess the accuracy of PDM, we treat it as a clustering algorithm. This is because a direct evaluation of the extracted topics and representative reviews involves manually processing a large number of complaints and summarizing the multiple-aspect information

of hidden defects, which can be inefficient and subjective.

Baselines

As baseline methods, we use LDA and K-Means algorithms to cluster different types of defects.

- We choose the LDA topic modeling method (Blei et al., 2003) as a baseline method because it has been widely used in opinion extraction and summarization tasks (Moghaddam and Ester, 2011).
- We also choose the K-Means clustering algorithm (MacQueen et al., 1967) as another baseline method because it is commonly used in text clustering (Steinbach et al., 2000).

The same vocabularies that have been used in PDM are applied as features in LDA and K-Means clustering for a fair comparison.

For each review, we first run PDM to calculate the most relevant defect using Equation (3.12), and then take that defect ID as the cluster label for the review. In this way, we can build clusters of reviews according to their cluster labels.

LDA clustering is implemented in a similar way. For each review, we compute the topic distribution θ during LDA inference, and then take the topic which has the largest probability for that review as its cluster label.

Metrics

We use precision, recall, and F1 (Zaki and Meira 2014) as the performance metrics which are commonly used in evaluating machine learning algorithms to assess the clustering quality of PDM. We use the model produced complaint groups as clusters (each cluster ID denoted as C_i), and ground-truth complaint groups as partitions (each partition denoted as T_j). $n_{ij} = |C_i \cap T_j|$ denotes the number of instances common to cluster C_i and partition T_j. Precision, recall, and F1 are calculated using Equations (3.15), (3.16), and (3.17). Here j_i denotes the partition containing the maximum number of points from C_i, n_i is the size of cluster C_i, and $|T_{j_i}|$ is the size of partition T_{j_i}.

Clustering Performance

The performance of the clustering algorithms is shown in Table 3.5.

$$Precision_i = \frac{1}{n_i} \max_{j=1} n_{ij} = \frac{n_{ij_i}}{n_i} \tag{3.15}$$

$$Recall_i = \frac{n_{ij_i}}{|T_{j_i}|} \tag{3.16}$$

$$F1_i = 2\frac{Precision_i \cdot Recall_i}{Precision_i + Recall_i} \tag{3.17}$$

From the results in Table 3.5 (where bold indicates best value), we can see:

- PDM outperforms LDA and K-Means in term of precision, recall, and F1 in most cases.
- The only exception is the precision of the Chevrolet data set. K-Means and lexical features seem to involve less noise in the Chevrolet case.
- The advantage of PDM is the most significant regarding the recall ratio.

Table 3.5: Clustering Performance of PDM, LDA, and K-Means

Data sets	**PDM**			**LDA**			**K-Means**		
	P(%)	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**
Toyota	**89.67**	**87.10**	**84.29**	80.77	52.82	59.48	78.90	66.63	68.19
Honda	**95.03**	**91.83**	**90.72**	86.07	36.77	45.93	90.05	60.29	61.91
Chevrolet	87.80	**86.90**	**84.53**	83.40	40.66	48.38	**94.23**	58.95	63.54

3.3.3 Consistency Analysis on Product Defects and Mitigating Measures on NHTSA Datasets

To further assess the effectiveness and usefulness of PDM, we match the vehicle defects identified for a vehicle brand by PDM with the mitigating measures (TSB and Recall) carried out by the manufacturers.

The review datasets on the vehicle models of three manufacturer produced during two time windows (2005-2009 and 2010-2015) are used.

During this evaluation, we manually match the vehicle defects identified by PDM with the TSB/Recall records in the NHTSA database using SQL queries. For each defect, we consider

it matches a TSB/Recall if it has the same vehicle model and component as that TSB/Recall record, and has a description similar to the TSB/Recall record.

The consistency metrics in Table 3.6 indicate related TSB or recall actions have been made by manufacturers or NHTSA for most of the defects identified by PDM. The greatest consistency occurs in the Chevrolet 2005-2009 dataset, while the least consistency occurs in the Toyota 2005-2009 dataset. The defects identified by PDM have a lower matching level on this Toyota dataset.

Table 3.6: Consistency between Identified Defects and TSB/Recall Records

Years	**Honda**	**Toyota**	**Chevrolet**
2005-2009	73.33%	66.67%	100.00%
2010-2015	73.33%	80.00%	86.67%

3.3.4 Predictive Analysis on NHTSA Datasets

Results from the above analysis can raise a red flag for manufacturers or government regulators to pay close attention to the potential issues, and plan for defect mitigating measures, such as Product Recall or TSB. To simulate this process, we conduct another experiment to predict Product Recalls or TSBs by applying a logistic regression model using features produced by PDM and a baseline method (LDA). Specifically, we use these approaches to produce semantic representations of complaints as their semantic features, which are used for mitigation measure prediction, following the evaluation methods of Abrahams et al. (2015) and Blei et al. (2003). This experiment shows that PDM can capture better semantic features of complaints for mitigating measure prediction. We describe the experiment in the following sub-section.

Predictive Experiment

Following the methods in Abrahams et al. (2015), we extract text style features, such as word count, fog readability index, and average word length, as the first group of independent variables. The other features, including social information, sentiment, and component features in Abrahams et al. (2015), are not used since they do not apply to the NHTSA dataset. For example, no social information exists in this dataset, and sentiment features are not important as almost all the NHTSA complaints are negative reviews. The component features are already considered in defect probability features. Since we are interested in how semantic features such as defect information correlates to mitigating measures, we apply PDM to extract defects from the complaints. Next, we compute the probabilities of defects given a complaint as the second group of independent variables using posteriors from Equation (3.3). In fact, these defect probability features reflect the information of all

defect entities such as model, component, symptom, and incident date, and thus can be a representation of a complaint in the abstract defect space.

For comparison, we pick LDA as a baseline method, as it is a common method in obtaining semantic representations of documents (Blei et al. 2003, Heinrich 2008). To extract LDA topic features from complaints, we set the same number of topics N as the defect count in PDM, run LDA, and create an N-dimensional vector for each complaint, in which each component is the probability of the corresponding topic given that complaint.

In summary, the text style features are used as common features in both methods, but the PDM and LDA methods create different semantic features for NHTSA complaints in this experiment.

To show that PDM can calculate more accurate defect distributions over reviews in predicting product recalls, we compare PDM with standard LDA in terms of F1. For both algorithms, the number of topics is between 10 and 50, which increases by 10 each time. We first extract the topics (or defects in PDM), and then compute the topic (defect) probabilities given a review and use the three text style features to predict mitigating measures. We use logistic regression as the classifier following Equations (3.18) and (3.19), and randomly oversample the dataset to resolve the imbalanced data issue (8763 records without mitigation vs. 977 records with Recall or TSB).

$$P(C_1|\phi) = \sigma(W^T\phi) \tag{3.18}$$

$$P(C_2|\phi) = 1 - P(C_1|\phi) \tag{3.19}$$

Predictive Performance

We plot the average F1 scores for different numbers of topics or defects in Figure 3.2. The document representation extracted by PDM has significant advantage over that created by LDA, which indicates that PDM is more capable of capturing semantic features for defect mitigation prediction.

3.3.5 Qualitative Evaluation on Amazon Datasets

Experiments

We conduct two case studies to show the generalizability of the PDM by using the negative reviews of electronics (Google Chromecast) and tablet products (Amazon Kindles). In order

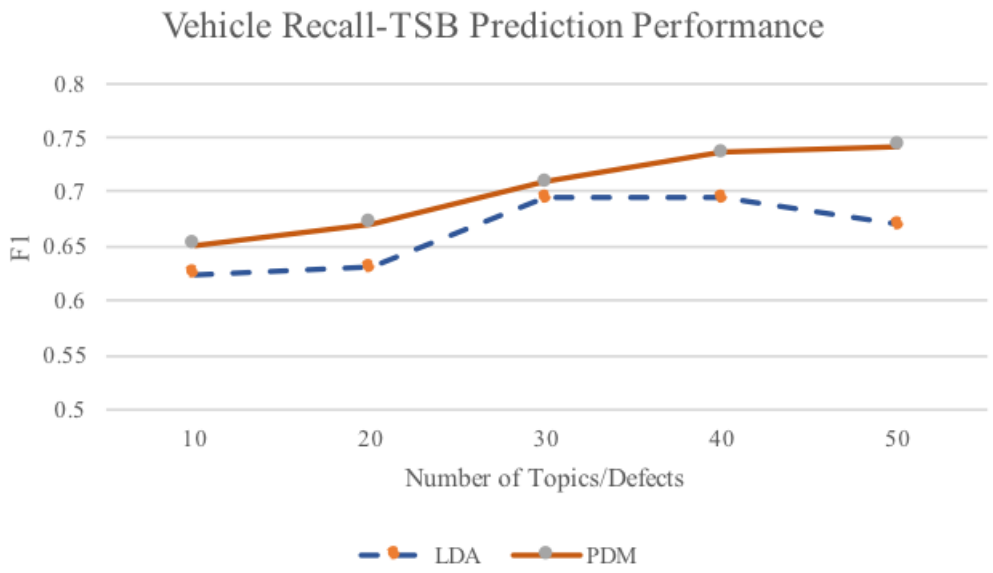


Figure 3.2: Vehicle Recall Prediction Performance of PDM and LDA, measured by Weighted F1

to apply PDM to these datasets efficiently, we pre-process the data to extract the complaint entities required by PDM. Then we run PDM at both the review level and the sentence level, and compare their topic coherence. We also compare topic readability and coherence against unigram and phrase symptom topics. The topics extracted from these datasets at review level are presented in Tables 3.7 and 3.9, while the topics extracted at sentence level are shown in Tables 3.8 and 3.10. The irrelevant words in those topics are underlined.

The entity extraction work focuses on the "Component" entity since other entities including "Model", "Date", and "Symptom" are relatively easy to extract from corresponding fields in these datasets. In order to extract Component entities from reviews with low cost, we use an unsupervised aspect extraction method proposed by Moghaddam and Ester (2012). First, all the nouns in reviews are recognized using POS tagging. Next, we apply frequent item set mining (Hu and Liu, 2004) to find high-frequency nouns or noun phrases, which are candidates of component words. Finally, the noise in these candidates are removed using a manually edited stop-word list. In this way, we can identify component words from each review. For both cases, the top 5 keywords of Component topic and top 10 keywords of each Symptom topic (in unigram or phrase) are identified. Irrelevant topic words are underscored.

We use the same phrase extraction method mentioned in Section 3.1 to get symptom phrases from review texts. To remove noise from the symptom description of reviews, we apply the same TF-IDF and POS filtering methods introduced in Section 3.1. Using the above pre-processing steps, we are able to extract word or phrase level entities needed by PDM.

Table 3.7: Top Defects of Google Chromecast Identified by PDM at Review Level

Model	**PDM at the Review Level**		
	Comp Unigram	**Symp Unigram**	**Symp Phrase**
Google Chromecast HDMI Streaming Media Player 2013	video	TV	Chrome browser
	streaming	work	smart TV
	stream	Chromecast	stream video
	browser	device	not stream
	phone	Google	watch video
		video	not support
		Netflix	not stream
		stream	Android device
		YouTube	watch Netflix
		APP	play video
	router	work	be compatible
	network	router	not compatible
	connection	Chromecast	wireless router
	setup	device	Chromecast work
	Internet	connect	watch movie
		Google	not stream
		TV	watch Netflix
		time	wireless network
		network	Chrome cast
		support	be unable
	cable	TV	watch movie
	laptop	Chromecast	Chrome browser
	video	work	smart TV
	computer	device	not connect
	connection	HDMI	buy TV
		video	Chromecast have
		cable	be disappointed
		Google	be connected
		laptop	watch Netflix
		computer	be TV

Table 3.8: Top Defects of Google Chromecast Identified by PDM at Sentence Level

Model	**PDM at the Sentence Level**		
	Comp Unigram	**Symp Unigram**	**Symp Phrase**
Google Chromecast HDMI Streaming Media Player 2013	stream streaming browser screen content	stream TV video chrome streaming browser Netflix screen YouTube content	Chrome browser not stream not support stream video cast tab stream content only stream be supported play video not play
	router network connection Internet setup	router network connection Internet setup wireless WiFi TV support	not connect be compatible wireless network wireless router not find not compatible Chromecast work new router Chromecast connect
	HDMI cable plug power port	TV HDMI plug cable power port laptop USB connect play	big screen watch Netflix full screen big TV watch movie be cheap be gadget connect laptop HDMI cable be screen

Table 3.9: Top Defects of Amazon Kindle Tablets Identified by PDM at Review Level

Model	**PDM at the Review Level**		
	Comp Unigram	**Symp Unigram**	**Symp Phrase**
Kindle Fire HD 8.9" Tablet 2012	charger battery port screen power	charge charger battery work problem port screen issue APP send	not charge not work hold charge be problem charge port not hold warranty expire no longer same issue screen work
Kindle Fire HD Tablet 2012	screen browser silk memory Internet	screen book browser APP download silk blue read tablet button	be issue no problem not support weak battery mayday button hold button battery run Kindle support not matter write review
Kindle Fire HDX 7" Tablet 2013	screen store keyboard camera display	screen APP book email video battery work play store read	screen freeze be case face camera watch movie USB charger not work battery drain include charger not respond be horrible

Table 3.10: Top Defects of Amazon Kindle Tablets Identified by PDM at Sentence Level

Model	**PDM at the Sentence Level**		
	Comp Unigram	**Symp Unigram**	**Symp Phrase**
Kindle Fire HD 8.9" Tablet 2012	battery charger port battery life power	battery charge charger port life problem work power issue keyboard	not charge not work charge port hold charge battery charge no longer warranty expire USB charger be tablet be case
Kindle Fire HD Tablet 2012	browser button case power silk	browser button case silk power volume Internet web work APP	hit button not work mayday button hold button be browser be annoy be slow not support not show press button
Kindle Fire HDX 7" Tablet 2013	screen display reader touch flicker	screen display reader work read touch APP blue tablet book	not work screen freeze lock screen screen work not access touch screen not play Google store not connect screen stop

Discussion

Unlike NHTSA complaints, many Amazon complaints discuss more than one issue. This may deteriorate topic coherence if we run PDM by taking each review as a document. As shown in Tables 3.7 and 3.9, one topic may have many irrelevant words when applying topic modeling at the review level.

In order to improve topic coherence, we run PDM at the sentence level, which takes each sentence as a document. Normally, one sentence discusses one issue. In this way, the sentences in a review will be split into multiple documents, which share the same "Model" and "Date" entities, but may have different "Component" and "Symptom" entities. The topics extracted at the sentence level are shown in the last three columns in Tables 3.8 and 3.10. The words or phrases which are considered as irrelevant regarding the topics they were assigned are highlighted by underscore. Comparing the topics extracted at the review level and the sentence level, we can see that topics extracted at sentence level have better coherence and fewer irrelevant words than topics extracted at review level.

Table 3.11 summarizes the relevance rate of the topic word(s) in the two datasets. Here, "C" stands for Component topics, while "S" denotes Symptom topics. The relevance rate in general is greatly improved when running the model at the sentence level, as opposed to at the review level, except for the symptom unigrams for the Google Chromecast dataset. The model achieves better topic coherence when running at the sentence level than at the review level.

Similar to the NHTSA dataset, we extract symptom topics in phrases by applying a dependency parser (Chen and Manning, 2014), and compare them to the unigram topics. After identifying irrelevant words in topics produced at the review level, we find more irrelevant words among phrase topics than unigram topics. This indicates that phrase topics may have worse coherence although they have better readability when one review complains about multiple defects. However, for unigram and phrase topics extracted at the sentence level, they have similar coherence, while phrase topics clearly have better readability.

Table 3.11: Relevance Rate of Topic Words/Phrases in Two Datasets

Datasets	**Topic relevance at Review Level**			**Topic relevance at Sentence Level**		
	C Unigram	**S Unigram**	**S Phrase**	**C Unigram**	**S Unigram**	**S Phrase**
Google Chromecast	86.67%	90.00%	66.67%	93.33%	86.67%	83.33%
Amazon Kindles	80.00%	70.00%	76.67%	100.00%	83.33%	93.33%

Chapter 4

Multiple-Facet Latent Dirichlet Allocation for Product Defect Discovery

As Chapter 1 mentioned, this research task aims at identifying and summarizing both symptoms and resolutions of product defects. In contrast to PDM, which focuses on what the defects look like, this task steps further to find how people attempted or proposed to fix the defects from a given dataset.

Two typical product defect reviews from the NHTSA vehicle complaint database and Apple laptop forum are shown in Figure 4.1. According to our observation, most of the sentences in those reviews can be divided into three types: background sentences which present background information such as the product model; sentences in bold which describe the symptoms of the problem; and sentences in italics which illustrate resolution efforts to diagnose and fix the issue. In addition, there are component words, which may occur in both symptom and resolution sentences, and which give focus to a part or subsystem of the product. Component, symptom, and resolution characterize product defects, and often occur as

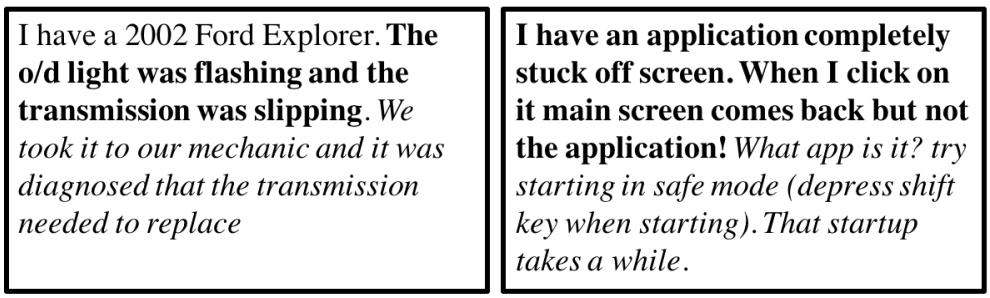


Figure 4.1: Examples of product defect reviews (**bold**: symptom, *italic*: resolution).

key entities in (or attributes of) product defect reviews.

The following considerations further make clear the distinctions between aspect opinion mining and product defect mining.

- "Component" (or "aspect") is important in both types of mining; defects are always related to some unit of a product.
- The "opinion" entity is not specific enough to describe the other essential types of defect information. For example, a product review usually contains information on "symptom" and "resolution", which state how the defect manifests, and the fixes (measures) users/mechanics tried, respectively. Instead of "aspect + opinion", an entity pattern such as "component + symptom + resolution" is needed for modeling defects. However, the "sentiment" entity becomes no longer important because most of the review words are negative.
- The entity location in customer reviews is different. Traditional aspect opinion mining classifies text into "aspect" and "opinion" entities at the word level. "Aspect" words and "opinion" words often occur as **word pairs** (e.g., "screen" and "clear" in the last example) (Moghaddam and Ester, 2012). But product defect mining usually needs to segment entities (especially "symptom" and "resolution") at the sentence level. Instead of appearing as word pairs, entities appear as **word groups** of varied sizes in reviews. For example, one review may have N_c "component" words, N_s "symptom" words, and N_r "resolution" words.
- In addition, words in "symptom" and "resolution" entities have quite diverse Part-Of-Speech (POS) labels (e.g., nouns, verbs, and adjectives) compared to the "opinion" entity of traditional methods, which is usually dominated by adjectives.

Based on the above observations, we model three correlated entities of product defect: Component, Symptom, and Resolution. We argue this model is effective to capture essential defect entities given a negative user review dataset. Here, Component words (e.g., "engine", "brake", "power train" in a vehicle complaint) point out the flawed units which may cause the defect. Symptom words (e.g., "acceleration", "hesitate", "fail") are a set of words that describe the symptoms of a product defect. In contrast, resolution words (e.g., "check", "accelerator", "reset") show how people attempt to diagnose and fix the issues. All the three entities of a defect are composed of words, which usually follow the multinomial distribution across all the review documents. As another observation, both symptom words and resolution words rely on the defective components. This will be an essential assumption for the Product Defect Latent Dirichlet Allocation (PDLDA) model devised for the problem.

Therefore, the objectives of PDLDA are:

- For each defect, the PDLDA model is supposed to produce a component topic, which is the core entity.

- The component topic should be followed by symptom topics and resolution topics which depend on it.
- These joint topics together provide comprehensive information about a defect.

This study makes the following contributions for product defect discovery from social media.

- We develop a PDLDA model, which jointly identifies the key entities of product defects ("Component", "Symptom", and "Resolution") from user reviews as interdependent three-dimensional topics.
- As opposed to multiple aspect LDA models, the proposed PDLDA model eliminates the dependence on word pairs. Existing LDA models (Moghaddam and Ester (2011); Wang and Ester (2014); Xue et al. (2015)) for aspect summarization usually require word pairs (e.g., an aspect word and an opinion word) during the training phase. In contrast, PDLDA accepts user reviews which have varying number of component, symptom, and resolution words.

In this chapter, Section 4.1 introduces the models for product defect discovery, including the algorithms for entity extraction, the design of the model, and the inference algorithm. Then, Section 4.2 presents the datasets, evaluation criteria, and baseline methods for the experiments. Finally, Section 4.3 demonstrates the results of the quantitative and qualitative evaluation, discusses them, and visualizes the PDLDA output.

4.1 Approach

4.1.1 Entity Extraction

When multiple types of entities, such as "aspect" and "opinion", are involved in an LDA model, we have to separate words of various types. For example, "screen" and "monitor" are aspect words while "clear" and "great" are opinion words. In our case, we need to find the words belonging to component, symptom, and resolution, given a review. There are mainly two strategies for this. Some researchers (Moghaddam and Ester, 2011; Zhan and Li, 2011; Wang and Ester, 2014; Xue et al., 2015) classify words into categories before entering them into LDA models, while others assign a type to each word inside the LDA model. The first strategy is more efficient for product defect mining, since words in one sentence usually belong to the same entity (except for the component entity).

The entity extraction module identifies words of 3 entities (i.e., component, symptom, and resolution) of each review, which are taken as input to the PDLDA model. Thus, the result of entity extraction has influence on the quality of the topics produced by the model. The entity extraction process has two steps: sentence segmentation and keyword filtering.

Lexicon-based methods are used to implement sentence segmentation. As stated in Section 4.2, there are at least two types of sentences in the text of a review: symptom sentences and resolution sentences. The NHTSA reviews also have a kind of ownership sentence, besides these two. The symptom words and resolution words dwell in those two types of sentences, respectively. Component words can be found in both types of sentences. We use lexicons (in the NHTSA datasets), or position in a forum thread (in the Apple forum and patient.info datasets), to distinguish them. Specifically, three lexicons (a noise, a resolution, and a component) are used to distinguish the three entities in the NHTSA dataset. The details can be found in Section 4.2.1. Assuming most sentences in a complaint review are symptom or resolution sentences, we identify the sentence type and entity type based on the word occurrence in the noise and resolution lexicons. The component lexicon is created using frequent item set mining as mentioned in Hu and Liu (2004), while the noise and the resolution word lexicons are built based on word frequency statistics in a small labeled data set. All the lexicons are reviewed and slightly adjusted by humans after creation to ensure the quality. Since this research concentrates more on the design of the topic model, rather than the entities segmentation, we used this light weight method to implement it. In the future, the entity extraction work can be improved by classification models.

Topic models are quite sensitive to noise such as stop-words. In this research, a filter based on POS tagging further removes non-informative words from candidate symptom and resolution words. Based on our observations, we assume the most informative words are nouns, verbs, and adjectives. Taking the symptom sentences as an example, we apply the Stanford POS tagger (Toutanova et al., 2003) to their words, keeping only nouns, verbs, and adjectives. The same filtering also happens to the resolution words. With these measures, we can provide high-quality observations (three entities of each review) to the PDLDA model.

4.1.2 PDLDA Model

As shown in Figure 4.2, PDLDA models key entities (component, symptom, and resolution) of defects with interdependent topics. There are 3 types of observations in the graphical model: *Component words* (W_c), *Symptom words* (W_s), and *Resolution words* (W_r). The topics assigned to these words are z, y, and x, respectively. We use two conditional probability matrices η and π to model the dependence between *symptom/resolution* and *component*. The notations used in this chapter are described in Table 4.1. The generation process of user reviews assumed by PDLDA is illustrated in Algorithm 2.

In the first step (first three loops), PDLDA determines the word distributions. For each component topic, a Multinomial distribution φ is sampled from the Dirichlet distribution $p(\varphi|\delta)$ as its word distribution. We also sample two Multinomials η and π from the Dirichlet distribution $p(\eta|\epsilon)$ and $p(\pi|\epsilon)$ as the corresponding symptom and resolution topic distributions which depend on this component topic. They are used as the dependence matrices, which model the dependency between component and symptom, and between component

Table 4.1: Definition of PDLDA Notations

K	number of topics
D	number of reviews (documents)
V_c, V_s, V_r	size of component, symptom, and resolution word vocabularies, respectively
N_c, N_s, N_r	numbers of component, symptom, and resolution words in a review, respectively
C, S, R	Component, Symptom, and Resolution words in the corpus
H	parameters of Dirichlet distributions, including $\alpha, \beta, \delta, \gamma, \epsilon$
φ, ψ, τ	parameters of the word distributions of different types of topics
θ, η, π	component topic distribution over documents, symptom topic distribution conditioned on component topics, and resolution topic distribution conditioned on component topics
λ_s, λ_r	Bernoulli distribution which decides whether a regular or a background topic is assigned to a word
t_s, t_r	whether a word is generated from a regular ($t = 0$) or a background topic ($t = 1$)
z_i, y_i, x_i	topic assigned to the i^{th} component, symptom, or resolution word
$z^{\neg i}$,$y^{\neg i}$, $x^{\neg i}$	topic assignment vector for various types of words in the corpus excluding the i^{th} word
$n_c^T(d,k)$	number of words in document d assigned to topic k; T means topic distribution
$n_c^W(k,v_c)$	number of component word v_c assigned to component topic k; W means word distribution
$n_s^T(k,l)$	number of symptom words assigned to topic l, whose corresponding component topic is k
$n_s^W(l,v_s)$	number of symptom word v_s assigned to symptom topic l
$n_r^T(k,m)$	number of resolution words assigned to topic m, whose corresponding component topic is k
$n_r^W(m,v_r)$	number of resolution word v_r assigned to resolution topic m

Algorithm 2: Review generation process of PDLDA

for *each component topic* **do**
  Sample word distribution $\varphi \sim Dirichlet(\delta)$
  Sample dependent topic distribution $\eta \sim Dirichlet(\epsilon)$ for *symp* entities
  Sample dependent topic distribution $\pi \sim Dirichlet(\epsilon)$ for *reso* entities
end
for *each symptom topic, including the background symptom topic* **do**
  Sample word distribution $\psi \sim Dirichlet(\beta)$
end
for *each resolution topic, including the background resolution topic* **do**
  Sample word distribution $\tau \sim Dirichlet(\gamma)$
end
for *each review* d **do**
  Sample $\theta_d \sim Multinomial(\alpha)$
  for *each* W_c *of* d **do**
    Sample a *comp* topic $z \sim Multinomial(\theta_d)$
    Sample $W_c \sim Multinomial(\varphi_z)$
  end
  Draw a *comp* topic z' from the set of z sampled above
  for *each* W_s *of* d **do**
    Sample a $t_s \sim Bernoulli(\lambda_s)$
    if $t_s = 0$ **then**
      Sample a *symp* topic $y \sim Multinomial(\eta_{z'})$
      Sample $W_s \sim Multinomial(\psi_y)$
    else if $t_s = 1$ **then**
      Sample $W_s \sim Multinomial(\psi_B)$
  end
  for *each* W_r *of* d **do**
    Sample a $t_r \sim Bernoulli(\lambda_r)$
    if $t_r = 0$ **then**
      Sample a *reso* topic $x \sim Multinomial(\pi_{z'})$
      Sample $W_r \sim Multinomial(\tau_x)$
    else if $t_r = 1$ **then**
      Sample $W_r \sim Multinomial(\tau_B)$
  end
end

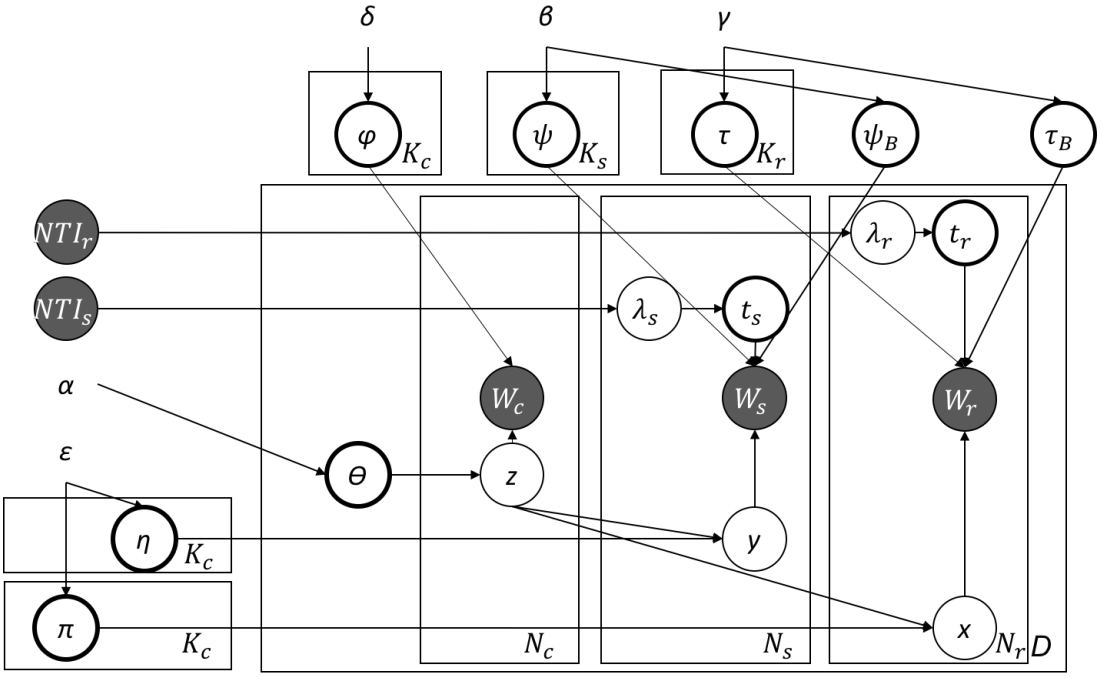


Figure 4.2: Plate Notation of PDLDA Model

and resolution, respectively. Multinomials ψ and τ are sampled as word distribution of each symptom and resolution topic (both regular and background topics), respectively. For example, ψ represents the word distribution of the K_s regular symptom topics, while a special ψ_B is the word distribution of the background symptom topic. We design background topics in the PDLDA model to remove the less-informative words (e.g., "vehicle", "run", and "show" in the NHTSA dataset) from the regular symptom and resolution topics. Specifically, during the Gibbs sampling process, those less-informative words, including the most common words in the symptom or resolution sentences, are more probable to be assigned a background topic; their sampling process is introduced later. Thus, they have low probability to occur in the regular symptom/resolution topics, which will reduce the noise in topics.

The topic assignment is determined in the remaining step (final loop). Given a review d, we first sample a component topic z for each of the N_c component words in a document d. The component topic sampling is the same as with standard LDA. As a result, we get a set of z, $Z_d = \{z_1, z_2, \cdots, z_{N_c}\}$. Then, one component topic z' is randomly drawn from this component topic set Z_d. We draw z' before the sampling of each symptom y and resolution topic x, since we assume they depend on this z'. In particular, y is sampled from $p(y|z')$, and x is sampled from $p(x|z')$, which are two multinomial distributions assuming the related component topic is z'. Next, to choose a symptom topic for each symptom word, we first identify whether it is a *background* or a *regular* symptom word. This indicator t_s is drawn from a Bernoulli distribution λ_s. If the word is a regular symptom word (if $t_s = 0$), we sample

a symptom topic y from Multinomial $p(y|\eta_{z'})$ for each symptom word W_s, then sample the word from Multinomial $p(W_s|\psi_y)$; otherwise (if $t_s = 1$), the background symptom topic is assigned to the word, and we sample it from Multinomial $p(W_s|\psi_B)$. The same sampling process is used during the generation of each resolution word W_r.

A special feature of the PDLDA model is the presence of two switch variables t_s and t_r, which are designed to reduce overlap between different symptom (or resolution) topics. In particular, we give the most common words high probability of being assigned into a background symptom or a background resolution topic. The TF-IDF value of a word is used to help decide whether it is a background or regular word. Specifically, we draw two indicators t_s and t_r from Bernoulli distributions λ_s and λ_r, respectively. The parameter of the Bernoulli distribution is initialized by the normalized TF-IDF value of that word, in particular $P(t_s = 0) = normalized\ TF * IDF$.

Using this sampling strategy, especially the sampling of a component topic for related symptom and resolution topics, we eliminate a key drawback of the existing aspect mining LDA models (Moghaddam and Ester, 2011; Wang and Ester, 2014; Xue et al., 2015), which can only take word pairs as model input and the signal to identify entity dependency. These methods reply on word pairs to capture dependency between aspect and opinion words. In particular, the opinion topic sampling for a opinion word depends on the aspect topic assigned to the aspect word in the same word pair[1]. However, for user reviews discussing product defects, in many cases the dependency between component and symptom (or resolution) can hardly be captured by the dependency parsing rules used by those models (e.g., ACOMP, AMOD, NSUBJ, DOBJ, CONJ_AND, COP, NN, and NEG introduced in Moghaddam and Ester (2012)). For example, the last sentence of the first review in Figure 4.1 is "... that the transmission needed to replace.". In this sentence, a component word is "transmission", and a resolution word is "replace". The dependency between those words cannot be identified using the above parsing rules, but PDLDA can learn this dependency by sampling the component topic for the resolution word. In addition, this feature also allows PDLDA to directly take word groups, e.g., a group with N_c "component" words, N_s "symptom" words, and N_r "resolution" words extracted from a review, as model input.

4.1.3 Model Inference

We will now provide the details of the inference of the proposed model which uses collapsed Gibbs sampling.

The inference process starts with the joint distributions of the 3 types of topics. They will be used in the inference of Gibbs updating rules, which is the key of Gibbs sampling. The

[1]This is the main reason why these models only accept word pairs as input.

joint distribution of component topic z and component word W_c is:

$$\begin{aligned} P(z, W_c|\alpha, \delta) &= P(W_c|z, \delta) \cdot P(z|\alpha) \\ &= \int P(W_c|z, \phi)P(\phi|\delta)d\phi \cdot \int P(z|\theta)P(\theta|\alpha)d\theta \\ &= \prod_{k=1}^{K_c} \frac{B(\phi_k + \delta)}{B(\delta)} \cdot \prod_{d=1}^{D} \frac{B(\theta_d + \alpha)}{B(\alpha)} \end{aligned} \tag{4.1}$$

Here, $B()$ is the multinomial beta function, which is denoted as Equation (4.2). $\Gamma()$ is the Gamma function (Equation (4.3)), which is a factorial function in LDA.

$$B(\alpha) = \frac{\prod_{i=1}^{|\alpha|} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{|\alpha|} \alpha_i)} \tag{4.2}$$

$$\Gamma(n) = (n-1)! \tag{4.3}$$

Similarly, the joint distribution of symptom topic and words is:

$$\begin{aligned} P(y, W_s|z, \beta) &= P(y|z, \epsilon) \cdot P(W_s|y, z, \beta) \\ &= \int P(W_s|y, \psi)P(\psi|\beta)d\psi \cdot \int P(y|\eta)P(\eta|\epsilon)d\eta \\ &= \prod_{k=1}^{K_s} \frac{B(\psi_k + \beta)}{B(\beta)} \cdot \prod_{z=1}^{K_c} \frac{B(\eta_z + \epsilon)}{B(\epsilon)} \end{aligned} \tag{4.4}$$

The joint distribution of resolution topic and words is:

$$\begin{aligned} P(x, W_r|z, \gamma) &= P(x|z, \epsilon) \cdot P(W_r|x, z, \gamma) \\ &= \int P(W_r|x, \tau)P(\tau|\gamma)d\tau \cdot \int P(x|\pi)P(\pi|\epsilon)d\pi \\ &= \prod_{k=1}^{K_r} \frac{B(\tau_k + \gamma)}{B(\gamma)} \cdot \prod_{z=1}^{K_c} \frac{B(\pi_z + \epsilon)}{B(\epsilon)} \end{aligned} \tag{4.5}$$

The Gibbs updating rules discussed below show how PDLDA assigns topics to the component, symptom, and resolution words in a review.

According to Equation (4.1), the Gibbs updating rule (Equation (4.6)) for component words

is:

$$\begin{aligned} P(z_i = k|z^{\neg i}, C, H) &= \frac{P(z, W_c|H)}{P(z^{\neg i}, W_c^{\neg i})P(W_c|z^{\neg i}, W_c^{\neg i}, H)} \\ &\propto \frac{P(z, W_c|H)}{P(z^{\neg i}, W_c^{\neg i}|H)} \\ &\propto \frac{n_c^T(d,k) + \alpha_k - 1}{\sum_{k'=1}^{K}(n_c^T(d,k') + \alpha_{k'}) - 1} \\ &\cdot \frac{n_c^W(k, v_c) + \delta_{v_c} - 1}{\sum_{v'=1}^{V_c}(n_c^W(k,v') + \delta_{v'}) - 1} \end{aligned} \tag{4.6}$$

Assuming the corresponding component topic of symptom word i is k, the Gibbs updating rule (Equation (4.7)) for symptom words is :

$$\begin{aligned} P(y_i = l|z, y^{\neg i}, S, H) &= \frac{P(y, W_s|z, H)}{P(y^{\neg i}, W_s^{\neg i}|z, H)P(W_s|y^{\neg i}, W_s^{\neg i}, z, H)} \\ &\propto \frac{P(y, W_s|z, H)}{P(y^{\neg i}, W_s^{\neg i}|z, H)} \\ &\propto \frac{n_s^T(k,l) + \epsilon_l - 1}{\sum_{l'=1}^{K}(n_s^T(k,l') + \epsilon_{l'}) - 1} \\ &\cdot \frac{n_s^W(l, v_s) + \beta_{v_s} - 1}{\sum_{v'=1}^{V_s}(n_s^W(l,v') + \beta_{v'}) - 1} \end{aligned} \tag{4.7}$$

Assuming the corresponding component topic of resolution word i is k, the Gibbs updating rule (Equation (4.8)) for resolution words is:

$$\begin{aligned} P(x_i = m|z, x^{\neg i}, R, H) &= \frac{P(x, W_r|z, H)}{P(x^{\neg i}, W_r^{\neg i}|z, H)P(W_r|x^{\neg i}, W_r^{\neg i}, z, H)} \\ &\propto \frac{P(x, W_r|z, H)}{P(x^{\neg i}, W_r^{\neg i}|z, H)} \\ &\propto \frac{n_r^T(k,m) + \epsilon_m - 1}{\sum_{m'=1}^{K}(n_r^T(k,m') + \epsilon_{m'}) - 1} \\ &\cdot \frac{n_r^W(m, v_r) + \gamma_{v_r} - 1}{\sum_{v'=1}^{V_r}(n_r^W(m,v') + \gamma_{v'}) - 1} \end{aligned} \tag{4.8}$$

As mentioned before, the model decides whether a symptom (or resolution) word should be assigned a background or regular symptom (or resolution) topic by sampling from a Bernoulli distribution and taking the word's normalized TF-IDF value as prior. Here, the augmented term frequency shown in Equation (4.9) is used to prevent a bias towards longer documents. Also, $f_{t,d}$ stands for the word frequency of term t in document d, and N is the number of documents in the entire collection. Then the TF-IDF value is normalized for each review

and taken as the parameter of the Bernoulli distribution λ_s or λ_r, from which we sample the type (background or regular) of that word.

$$TF/IDF(t,d) = (0.5 + 0.5\frac{f_{t,d}}{max\{f_{t',d:t'\in d}\}}) \cdot \log\frac{N}{|\{d \in D : t \in d\}|} \tag{4.9}$$

4.1.4 Implementation and Parameter Setting

PDLDA is implemented in Java to ensure the performance. In practice, we use symmetric priors for the Dirichlet distribution parameters $\alpha, \beta, \delta, \gamma$, and ϵ, and set them with empirical values $\alpha = 0.01$, $\beta = \gamma = \delta = \epsilon = 0.001$.

4.2 Data

In this experiment, we use the complaint database of NHTSA[2] and problem discussion threads in the official forum of Apple[3]. Besides the vehicle and computer defects, we are excited to see that the PDLDA model can discover human disease symptoms and treatment from patient discussions. The patient discussion threads in the patient.info website[4] are collected for that. Products and human diseases which received the most complaints (or discussions) are selected to create the datasets.

Two types of datasets are built for different experiments. For qualitative evaluation and topic coherence evaluation (e.g., PMI), labels are not necessary; thus we construct large datasets shown in Table 4.2. However, for the other experiments such as document clustering, human annotation results are needed as ground truth. Smaller datasets, such as the first two datasets in Table 4.3, are created and manually tagged for this evaluation. The reviews in each dataset are split into multiple clusters according to the defect or sickness they present. The labeling and splitting of the FORD Focus dataset and APPLE MacBook dataset in Table 4.3 is done by human. A special case is the patient.info dataset. We can use the same dataset for both qualitative evaluation and the document clustering experiment because we take the sub-forum name of each discussion thread as its clustering label. This means we don't need to manually label that dataset to acquire ground truth. Thus we have one quite large dataset for clustering performance evaluation in contrast to the other two. A review dataset is constructed for each product model for simplicity. When we create the NHTSA

[2] `https://www-odi.nhtsa.dot.gov/downloads/flatfiles.cfm`

[3] `https://discussions.apple.com`

[4] `https://patient.info/forums`

Table 4.2: Datasets for Qualitative and Topic Coherence Evaluation

Product	**Number of reviews**
FORD FOCUS	10,099
APPLE MacBook	2,098
Patient.info	43,928

Table 4.3: Datasets for Clustering Performance Evaluation

Product	**Number of reviews**	**Number of clusters**
FORD FOCUS	1,941	4
APPLE MacBook	413	5
Patient.info	43,928	10

and Apple forum datasets in Table 4.3, reviews with vague cluster assignment (e.g., a review can be assigned to multiple clusters) were removed to avoid any ambiguity. Small clusters also were removed to avoid unbalanced data issues.

4.2.1 Preprocessing

As mentioned in Section 4.1.1, lexicon-based and structure-based methods are used to classify review sentences and then extract the three entities of each review.

The frequent item set mining approach is used to extract components in all the datasets, by following the method introduced in Hu and Liu (2004). Figure 4.3 shows the main process of component entity extraction. Component entities are usually nouns or noun phrases in review sentences. Therefore, a POS tagging is performed first to identify nouns in the review text. Next, we use the FP-Growth algorithm (Han et al., 2000) to identify the most frequent nouns and noun phrases. Then, the redundancy pruning method introduced in Hu and Liu (2004) is applied to remove redundant unigrams. The remaining frequent component words and phrases will be put into a component lexicon. Finally, we manually review and tune the lexicon, which can be used to extract component entities from review text.

Regarding the extraction of symptom and resolution entities, different strategies are used on various datasets.

- The lexicon-based method is applied to the NHSTA datasets. There are three types of sentences in the NHTSA reviews: ownership, symptom, and resolution, which mix in the "CDESCR" column of the NHTSA complaint records. To create the needed lexicons, we go through the sentences of a small dataset which has 200 reviews of different vehicle models, find the feature words (e.g., "buy", "purchase", and "own") of ownership sentences, and find the feature words (e.g., "fix", "remedy", and "resolve")

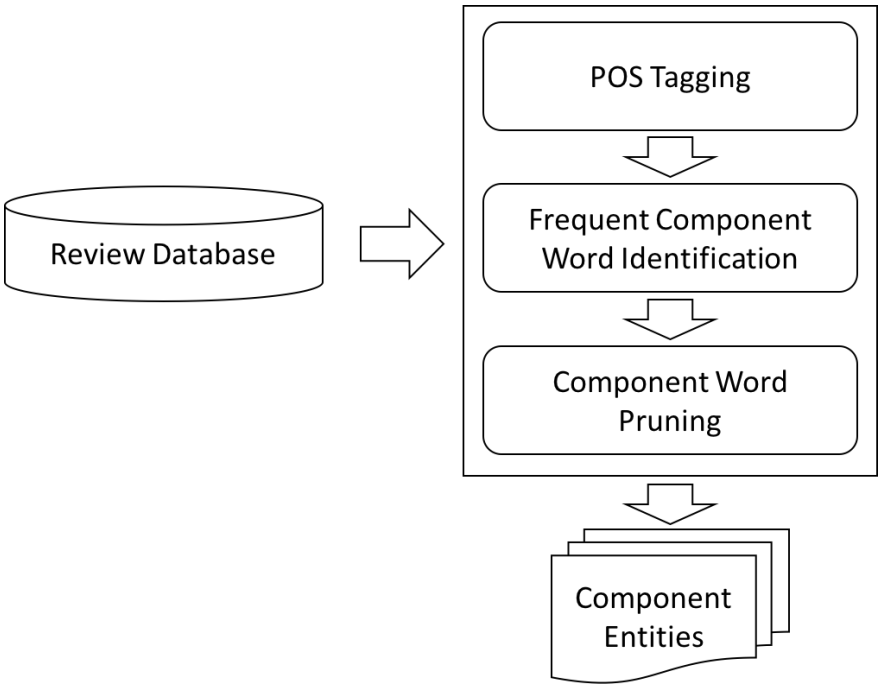


Figure 4.3: Extract Component Entities by Frequent Item Set Mining

of resolution sentences. In this way, two lexicons are created, which will help classify the sentence type according to the lexicon word occurrence.

- For the Apple forum and the patient.info datasets, we rely on the thread structure to separate them. Specifically, each forum thread is taken as a product review. Then, we take the sentences in the first post as candidate symptom sentences and those in the "most recommended solution" post as candidate resolution sentences.

4.2.2 Evaluation Criteria

Both quantitative and qualitative evaluations are conducted for the evaluation of the proposed PDLDA model.

- *Topic Coherence*: The Pointwise Mutual Information (PMI) metric (Newman et al., 2010) is chosen to measure the coherence level of the words within a topic. Literally, two words occurring in the same topic may have high coherence score if they have high co-occurrence in the original document collection. On the other hand, two words may have low coherence score if they seldom appear in the same context. The PMI score which reflects coherence level of topic words is calculated by Equations (4.10) and (4.11). Here, w_i and w_j are words among the top-N words of a topic. $p(w_i, w_j)$

stands for the co-occurrence probability of w_i and w_j in a document collection, while $p(w_i)$ and $p(w_j)$ are the probability of the single words w_i and w_j themselves.

- *Entity Clustering Accuracy*: Precision, Recall, and F-Measure which have been used in Moghaddam and Ester (2012) are selected as the criteria to measure entity (e.g., symptom and resolution) clustering performance. Specifically, we followed the method presented in Zaki et al. (2014) when calculating those metrics. The same metrics have been used in the evaluation of PDM.
- *Qualitative Evaluation*: The top defects of vehicle models are identified from reviews as joint topics.

$$PMI - Score(w) = mean\{PMI(w_i, w_j), ij \in 1...N, i \neq j\} \tag{4.10}$$

$$PMI(w_i, w_j) = log\frac{p(w_i, w_j)}{p(w_i)p(w_j)} \tag{4.11}$$

4.2.3 Baseline Methods

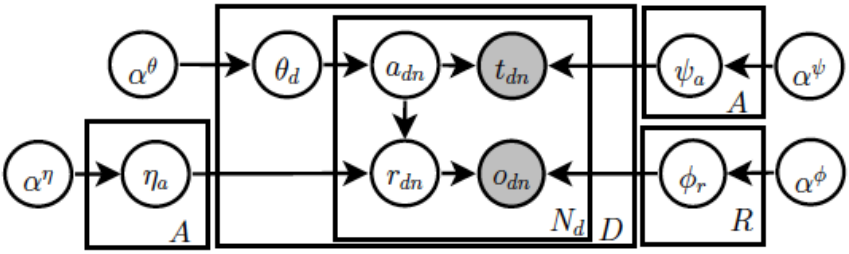


Figure 4.4: Plate Notation of ILDA (Moghaddam and Ester, 2011)

- *Standard LDA* (Blei et al., 2003): After entity extraction using the methods in Section 3.2, we run standard LDA on the words of three entities separately to get three kinds of topics.
- *ILDA* (Moghaddam and Ester, 2011): Figure 4.4 shows the graphical model of ILDA, which first generates an aspect word t_{dn} then generates the corresponding opinion word o_{dn}. The sampling of opinion topic r_{dn} relies on the aspect topic a_{dn}. In its preprocessing step, ILDA uses a dependency parser (Chen and Manning, 2014) to

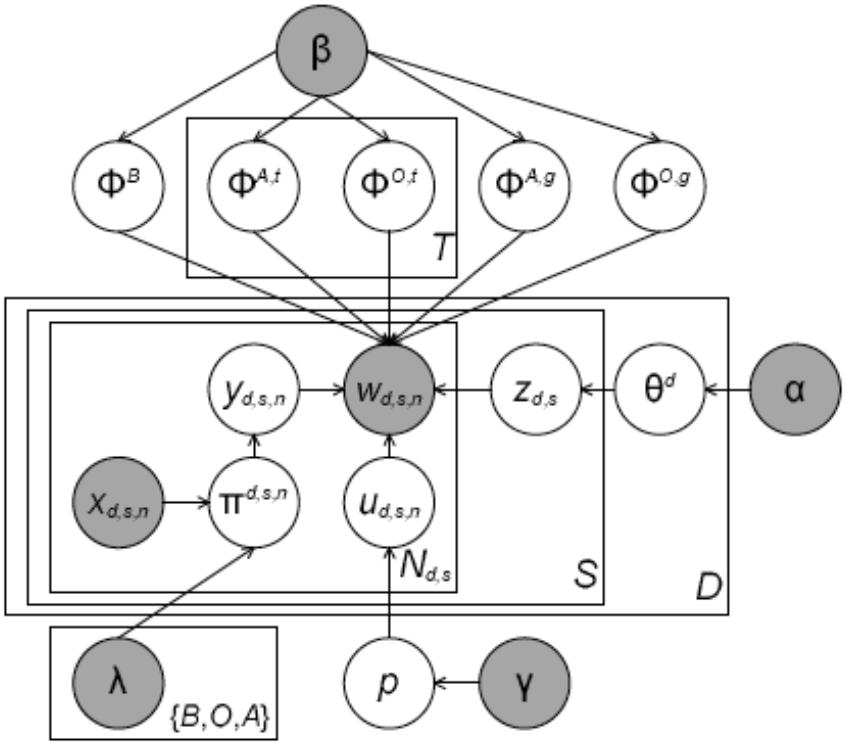


Figure 4.5: Plate Notation of MaxEnt LDA (Zhao et al., 2010)

parse the review text into a set of phrases. Each phrase is a pair of ⟨aspect word, opinion word⟩. To apply ILDA to product defect discovery, we get the component (aspect) and the symptom (opinion) topics by taking symptom sentences as input, and get the component (aspect) and the resolution (opinion) topics from resolution sentences. A dependency parser (Chen and Manning, 2014) is applied when extracting the ⟨aspect word, opinion word⟩ pairs. Most of the dependency patterns introduced in Moghaddam and Ester (2012), including "AMO", "NSUBJ", "DOBJ", "ADVCL", "ADVMOD", "AUXPASS", "COP", "NN", and "NEG", have been used for phrase extraction.

- *MaxEnt LDA* (Zhao et al., 2010): Shown in Figure 4.5, MaxEnt LDA incorporates a maximum entropy mechanism into the topic model to discriminate the types (e.g., background, aspect, and opinion) of each word based on its feature vector $X_{d,s,n}$. The feature vector comprises two types of features: (1) lexicon features such as the previous, the current, and the next words; (2) POS label features, such as the previous, the current, and the next POS labels. There are two special switch variables in this model which decide word type during topic and word sampling. $y_{d,s,n}$ is sampled from a multinomial distribution $\pi^{d,s,n}$, and it determines whether the word $w_{d,s,n}$ is a background, aspect, or opinion word. Another switch $u_{d,s,n}$ is sampled from a Bernoulli

distribution p, which determines whether $w_{d,s,n}$ is a general or aspect-specific word. We use the same symptom and resolution sentences used in ILDA to produce topics by MaxEnt LDA. The opinion lexicon in Hu and Liu (2004) and our own component lexicon which are introduced in Section 4.2.1 are used to label word types when training the Maximum Entropy model. In particular, the words appearing in the opinion lexicon are marked as opinion type, the words seen in the component lexicon are labeled as aspect type, and the others are treated as background type.

4.3 Evaluation Results

4.3.1 Performance of Review Clustering

In this evaluation, the symptom and resolution entities of reviews are clustered, then we measure the accuracy of the clustering. Each review has three types of entities: component, symptom, and resolution, as introduced in Section 4.1.1. When we apply PDLDA and the baselines, we extract these entities and sample topics for the entity words. The symptom and resolution entities of the reviews are clustered according to the topics assigned to the entity words. For example, when clustering symptom entities of reviews, we take the symptom topic ID assigned to the majority of symptom words of a review as its symptom cluster, following the method introduced by Heinrich (2008) and Moghaddam and Ester (2011). A similar clustering process occurs when the resolution entities of the reviews are clustered. Different topic models produce different topic sampling results, which lead to different clustering accuracy.

The clustering performance on the three datasets is shown in Tables 4.4, 4.5, and 4.6. We can see the precision and F1-Measure of PDLDA are better than the baselines, for both symptom and resolution clustering on all the datasets. MaxEnt LDA has better recall rate on the MacBook dataset. Since PDLDA jointly models the three entities, it leverages the influence of component entities when clustering dependent entities. Therefore, it outperforms the standard LDA. In addition, PDLDA extracts more reasonable words on a product defect than ILDA and MaxEnt LDA; thus it captures more important features when clustering review entities. The precision and scores in this experiment are not quite high, because we removed the component words, which are more distinctive for different defects, from the symptom and resolution entities. The remaining words in the symptom and resolution entities are more difficult to cluster and separate. Furthermore, the clustering performance on resolution entities is low for all the models, compared to the symptom entities. This reveals that the words in resolution sentences are more difficult to separate. We observe less diversity in the words describing problem fixing measures, in contrast to symptom words, especially when we exclude the component words.

Table 4.4: Review clustering Precision, Recall, and F-1 (%) on FORD Focus Dataset.

Dataset	**FORD Focus**					
Entity	**Symptom**			**Resolution**		
Method	**P(%)**	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**
LDA	69.09	55.47	56.35	68.72	25.00	36.55
ILDA	69.58	37.74	47.52	64.86	32.31	39.02
MaxEnt	64.38	26.53	23.77	73.62	25.94	22.69
PDLDA	**88.82**	**57.08**	**59.24**	**91.36**	**33.34**	**43.51**

Table 4.5: Review clustering Precision, Recall, and F-1 (%) on Apple MacBook Dataset.

Dataset	**APPLE MacBook**					
Entity	**Symptom**			**Resolution**		
Method	**P(%)**	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**
LDA	31.1	24.84	27.33	32.32	25.94	28.69
ILDA	30.31	28.27	29.65	30.79	23.13	25.11
MaxEnt	37.44	**33.84**	34.98	35.39	**31.21**	31.93
PDLDA	**57.7**	32.84	**35.29**	**49.62**	30.25	**33.69**

Table 4.6: Review clustering Precision, Recall, and F-1 (%) on patient.info Dataset.

Dataset	**Patient.info**					
Entity	**Symptom**			**Resolution**		
Method	**P(%)**	**R(%)**	**F1(%)**	**P(%)**	**R(%)**	**F1(%)**
LDA	43.72	45.97	44.45	31.17	29.98	30.32
ILDA	40.89	30.80	34.36	31.45	19.90	23.17
MaxEnt	41.19	41.30	40.52	30.88	30.47	30.39
PDLDA	**53.82**	**43.82**	**44.52**	**36.96**	**30.56**	**31.15**

A significance test has been carried out on the $F1$ scores in Tables 4.4, 4.5, and 4.6, using Equations (4.12) and (4.13). Here, A_1 and A_2 mean the accuracy of $Model_1$ and $Model_2$, for example, $F1$ of PDLDA and ILDA. N_1 and N_2 denote the number of instances used in the experiment. Variable d in Equation (4.13) stands for the difference between A_1 and A_2, while $Z_{\alpha/2}$ is the significance test coefficient, which is set to 1.96 for a 95% confidence level. The performance difference of two models is significant if the confidence interval doesn't contain 0.

$$\hat{\sigma_d} = \frac{A_1(1-A_1)}{N_1} + \frac{A_2(1-A_2)}{N_2} \tag{4.12}$$

$$CI = d \pm Z_{\alpha/2} \cdot \sqrt{\hat{\sigma}_d} \tag{4.13}$$

PDLDA has significant advantage over baselines in terms of $F1$ except in a few cases.

- MaxEnt Symptom and Resolution F1 on the Apple MacBook dataset.
- LDA Symptom F1 on the Patient.info dataset.

The potential reasons might include the following.

- As a complicated model, PDLDA may not have obvious advantage on small datasets (e.g., the MacBook dataset), which is also observed by Moghaddam and Ester (2012).
- For the patient.info dataset, we take the sub-forum name of each review as label, which hence is not directly labeled by researchers, thus the cluster label quality is lower than for the other two datasets.

4.3.2 Performance of Topic Coherence

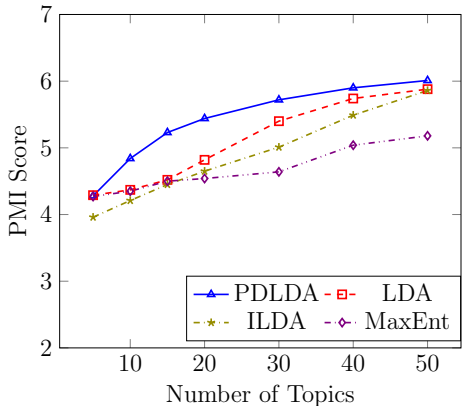
We can easily infer the word distribution of different topics based on the Gibbs updating rules in Equations (4.6), (4.7), and (4.8). Given a component, symptom, or resolution topic, its word probability can be computed using Equations (4.14), (4.15), or (4.16), respectively. These equations are derived from the second part of the Gibbs updating rules. Once the Gibbs sampling converges, the word distribution of a topic can be figured out using these equations.

$$\phi_{k,v_c} = \frac{n_c^W(k, v_c) + \delta_{v_c}}{\sum_{v'=1}^{V_c}(n_c^W(k, v') + \delta_{v'})} \tag{4.14}$$

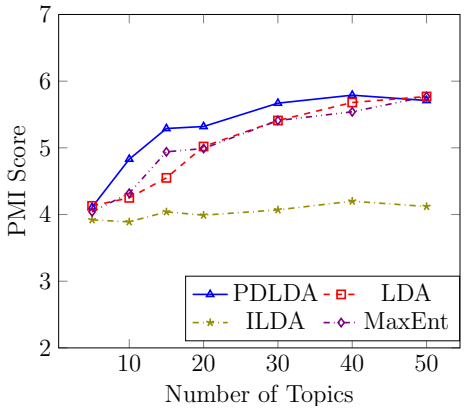
$$\psi_{l,v_s} = \frac{n_s^W(l, v_s) + \beta_{v_s}}{\sum_{v'=1}^{V_s}(n_s^W(l, v') + \beta_{v'})} \tag{4.15}$$

$$\tau_{m,v_r} = \frac{n_r^W(m, v_r) + \gamma_{v_r}}{\sum_{v'=1}^{V_r}(n_r^W(m, v') + \gamma_{v'})} \tag{4.16}$$

Once we get the words of different types of topics, we can apply Equation (4.10) to measure the PMI topic coherence scores. We measure and compare this metric when applying PDLDA and the baselines to the Focus 98-04, Patient.info, and MacBook datasets. The performance of various topic models are shown in Figures 4.6, 4.7, and 4.8.

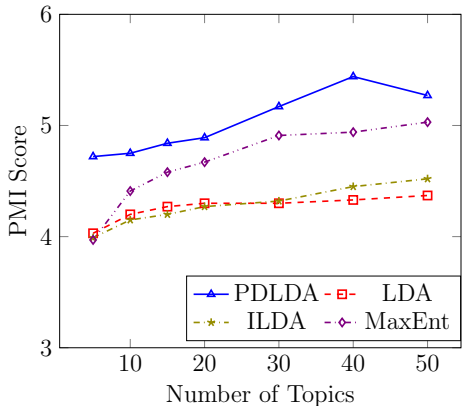


(a) Focus 98-04 Symptom

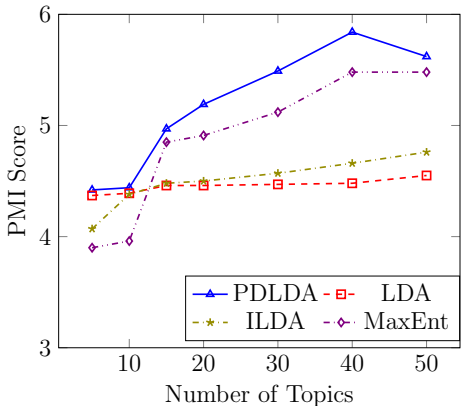


(b) Focus 98-04 Resolution

Figure 4.6: PMI coherence of symptom and resolution topics for FORD Focus.

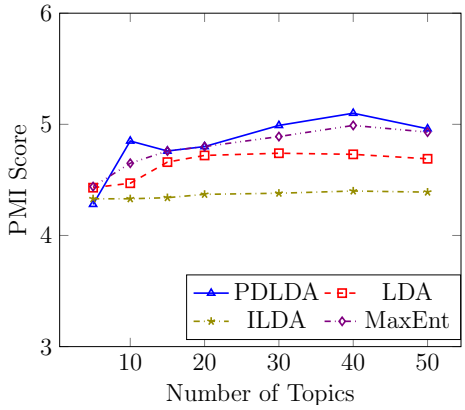


(a) MacBook Symptom

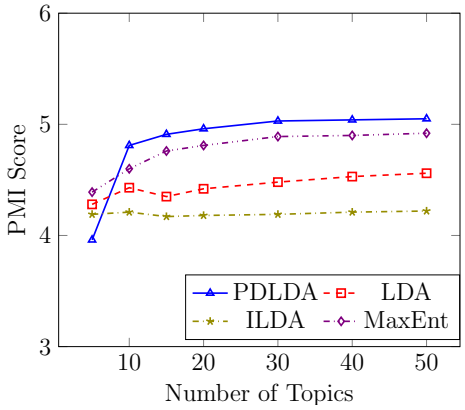


(b) MacBook Resolution

Figure 4.7: PMI coherence of symptom and resolution topics for APPLE MacBook.



(a) Patient.info Symptom



(b) Patient.info Resolution

Figure 4.8: PMI coherence of symptom and resolution topics for patient.info.

Since the sampling of component topics in PDLDA is the same as the standard LDA, they perform quite similarly on component topic coherence. Thus we only show the topic coherence of symptom and resolution topics. PDLDA outperforms the baseline methods in most cases, especially when the number of topics is over 15. Both the topic interdependency and entity identification mechanisms contribute to this. MaxEnt LDA has obvious advantage over the other two methods on the Patient.info and MacBook datasets; it benefits from incorporating prior knowledge (i.e., tagged word types) when identifying word types. Simple models such as standard LDA perform better in the FORD Focus dataset, and it works well for a small number of topics (e.g., *topic number* = 5), which matches the findings in Moghaddam and Ester (2012).

4.3.3 Qualitative Evaluation

While capturing critical defects of a vehicle model, PDLDA uses joint topics, composed of the most probable component, symptom, and resolution words, to show the defect information. In our output, each defect is led by its component topic, followed by relevant symptom and resolution topics. The component topic distribution $p(\theta_{d,k})$ in document k is determined by Equation (4.17). Given the component topic k, the most relevant symptom topic l can be obtained by sorting $\eta_{k,l}$, which is calculated by Equation (4.18). Similarly, the most relevant resolution topic m given component topic k is obtained by sorting $\pi_{k,m}$, which is calculated by Equation (4.19). The keywords of all types of topics can be figured out using the second part of the Gibbs updating rules. Since the component topic is decisive for a defect, the representative reviews of a defect can be obtained by sorting reviews by the $\theta_{d,k}$ value, assuming its component topic is k. The meaning of the variables in these equations can be found in Table 4.1.

$$\theta_{d,k} = \frac{n_{ct}(d,k) + \alpha_k}{\sum_{k'=1}^{K}(n_{ct}(d,k') + \alpha_{k'})} \tag{4.17}$$

$$\eta_{k,l} = \frac{n_{st}(k,l) + \epsilon_l}{\sum_{l'=1}^{K}(n_{st}(k,l') + \epsilon_{l'})} \tag{4.18}$$

$$\pi_{k,m} = \frac{n_{rt}(k,m) + \epsilon_m}{\sum_{m'=1}^{K}(n_{rt}(k,m') + \epsilon_{m'})} \tag{4.19}$$

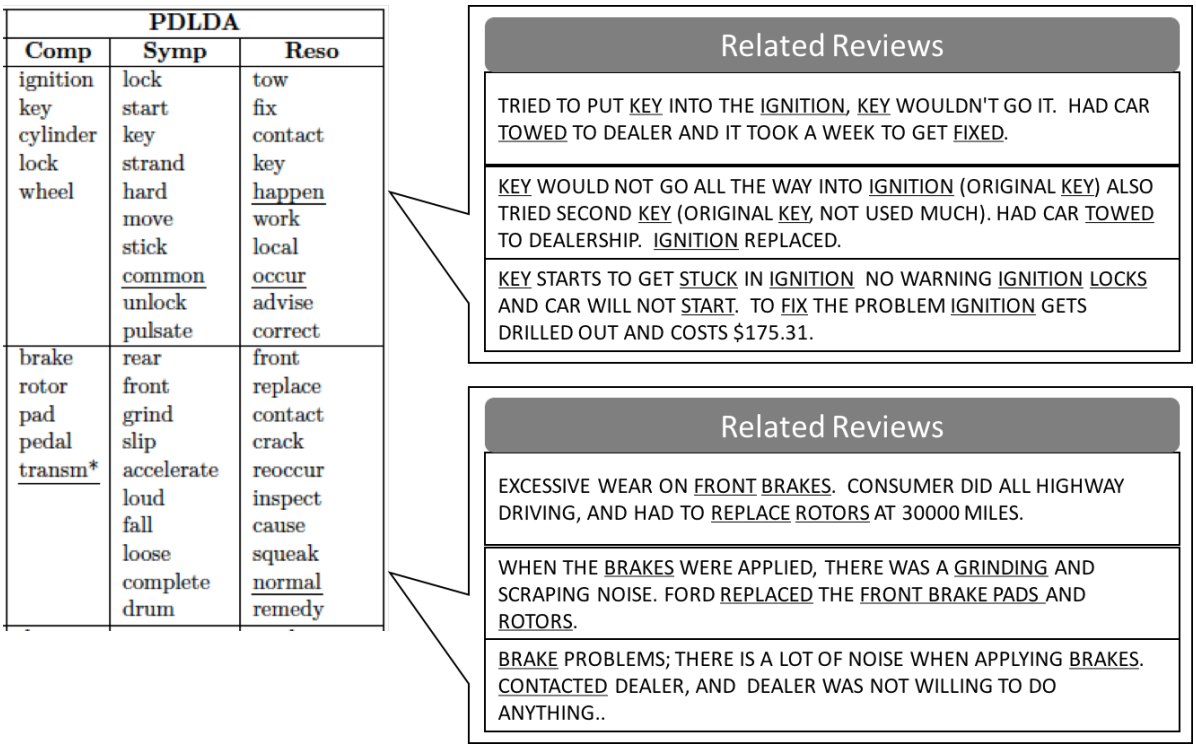
Table 4.7 shows joint topics extracted for two issues of 3 products. The top 5 words are shown for component topics, while the top 10 words are shown for symptom/resolution topics.

Looking at the joint topics produced by PDLDA, we can develop a good comprehension of the reported problems. Taking the ignition key problem of Ford Focus 98-04 in the third row as an example, the defective ignition part leads to a vehicle start problem and corresponding car towing. In another example in the 5th row, the flawed hard drive caused the crash of the Apple MacBook, which was solved by testing the hard drive and installing a new component. In summary, for each defect, users can locate the flawed units by reading the component topic, then learn what happens and how it is fixed by reading the dependent topics. The irrelevant words in the topics are underlined, through which we can see the topics generated by PDLDA have fewer irrelevant words.

Table 4.7: Example of topics extracted by LDA, ILDA, MaxEnt, and PDLDA. (irrelevant words are underlined; "transm*" in row 4 stands for "transmission")

Datset	LDA			ILDA			MaxEnt			PDLDA		
	Comp	**Symp**	**Reso**	**Comp**	**Symp**	**Reso**	**Comp**	**Symp**	**Reso**	**Comp**	**Symp**	**Reso**
FOCUS 98-04	ignition	turn	replace	key	ford	have	ignition	turn	replace	ignition	lock	tow
	key	start	call	focus	start	ford	key	steer	tow	key	start	fix
	cylinder	lock	tow	car	turn	take	lock	start	repair	cylinder	key	contact
	lock	key	aware	vehicle	key	numerous	turn	insert	call	lock	strand	key
	wheel	steer	contact	problem	put	recall	wheel	fail	turn	wheel	hard	happen
		unable	crack		get	resolve		put	drill		move	work
		fail	make		insert	receive		unable	install		stick	local
		put	work		go	ignition		place	require		common	occur
		insert	repair		ignition	get		park	strand		unlock	advise
		state	install		common	defective		stick	talk		pulsate	correct
	brake	brake	engine	noise	front	replace	brake	noise	replace	brake	rear	front
	rotor	work	check	brake	make	new	mile	apply	front	rotor	front	replace
	pad	problem	light	wheel	right	cause	pedal	grind	noise	pad	grind	contact
	pedal	start	mechanic	vehicle	cause	brake	consumer	hear	fail	pedal	slip	crack
	transm*	stop	drive	tire	left	last	stop	make	leak	transm*	accelerate	reoccur
		warranty	transmission		hit	bad		squeak	strut		loud	inspect
		day	door		loud	several		wear	bad		fall	cause
		purchase	cover		complete	turn		loud	turn		loose	squeak
		mile	warranty		pull	entire		mph	left		complete	normal
		apply	latch		other	break		squeal	grind		drum	remedy
MacBook	drive	mac	work	drive	use	use	drive	hard	click	drive	mac	mode
	file	apple	turn	thing	new	see	disk	external	open	file	hard	turn
	usb	show	system	pro	external	good	machine	old	safari	ssd	work	install
	backup	delete	delete	disk	other	support	ssd	erase	select	usb	device	hard
	disk	external	set	one	do	select	mode	reset	restart	backup	external	machine
		download	computer		hard	put		boot	erase		back	usb-c
		change	issue		connect	hard		order	safe		system	test
		book	option		try	buy		mount	paste		install	card
		back	app		thank	usb-c		work	imovie		MacBookpro	remove
		mail	make		few	few		format	quit		crash	die
	battery	MacBook	MacBook	monitor	use	use	battery	pro	charge	battery	charge	charge
	power	charge	plug	display	new	see	MacBook	charge	run	power	turn	reset
	charger	plug	pro	pro	external	good	power	plug	plug	charger	shut	support
	adapter	pro	work	port	other	support	charger	stay	occur	bluetooth	start	run
	charge	mbp	run	adapter	do	select	cycle	burn	nvram	monitor	slow	adapter
		turn	mac		hard	put		early	claim		computer	mbp
		work	cycle		connect	hard		die	mine		safari	replace
		close	charge		try	buy		flaw	recognize		usb-c	cycle
		old	end		thank	usb-c		respond	hear		good	external
		stop	replace		few	few		pad	discharge		black	put
Patient. info	hip	walk	recovery	side	left	use	walk	pain	pain	hip	walk	recovery
	knee	post	walk	hip	right	left	back	work	work	leg	thigh	replacement
	thyroid	exercise	exercise	leg	have	meet	exercise	good	good	muscle	crutch	joint
	face	surgeon	heal	eye	other	right	physio	well	better	knee	physio	physio
	blood	pain	post	replacement	new	dry	bed	hurt	recovery	weight	groin	swell
		thigh	replacement		total	put		lift	well		operate	thigh
		crutch	bone		upper	keep		limp	crutch		bone	revision
		physio	joint		swollen	wear		bad	lift		limp	consultant
		stick	longer		operate	new		operate	step		prednisone	injection
		joint	recover		move	sore		painful	best		operation	lift
	stress	start	make	anxiety	have	have	thought	anxious	fear	anxiety	panic	panic
	brain	med	feel	attack	bad	get	life	love	worry	heart	attack	meditation
	anxiety	side	worser	fear	get	cause	make	work	symptom	mind	die	fear
	mind	better	better	disorder	cause	few	feel	die	negative	body	beat	feeling
	thyroid	effect	work	depression	severe	last	control	worry	positive	chest	breathe	life
		feel	bad		constant	bad		fear	hard		anxious	talk
		stop	hope		suffer	same		cry	better		wrong	deep
		work	doc		experience	feel		scared	anxious		stop	rate
		bad	stop		develop	other		bad	worse		scared	technique
		worse	worry		experienced	first		good	strong		doctor	sound

The defects can be sorted by their component topic probability during demonstration. The probability of component topic k can be calculated by Equation (4.20), where $p(\theta_{d,k})$ is figured out by Equation (4.17).



PDLDA		
Comp	**Symp**	**Reso**
ignition	lock	tow
key	start	fix
cylinder	key	contact
lock	strand	key
wheel	hard	happen
	move	work
	stick	local
	common	occur
	unlock	advise
	pulsate	correct
brake	rear	front
rotor	front	replace
pad	grind	contact
pedal	slip	crack
transm*	accelerate	reoccur
	loud	inspect
	fall	cause
	loose	squeak
	complete	normal
	drum	remedy

Figure 4.9: The most relevant reviews of defects.

$$p(k) = \sum_{d=1}^{D} (\theta_{d,k} \cdot p(d)) \tag{4.20}$$

Furthermore, we can show the most relevant reviews for a defect. This can be implemented by sorting the reviews by the component topic probability of that defect, which can be figured out by Equation (4.17). Figure 4.9 shows an example of the most relevant reviews of the first two defects in Table 4.7. The top 5 component topic words, top 10 symptom topic words, and top 10 resolution topic words appearing in the related reviews are underlined.

As demonstrated above, PDLDA can effectively and accurately identify product defects and related solutions from online user reviews. Based on the extracted information, customers can make their purchase decisions, manufacturers can improve their products and gather user requirements, and governments can conduct administrative actions.

[5] `http://128.173.49.55/cardefects/src/`

4.3.4 Visualization of Topics

With the help of our colleagues, especially Brandon Fan, we built a website[5] to visualize the results of PDLDA on the NHTSA datasets. This website can help customers learn the existing defects of various vehicle models. The front page of the website is shown in Figure 4.10.

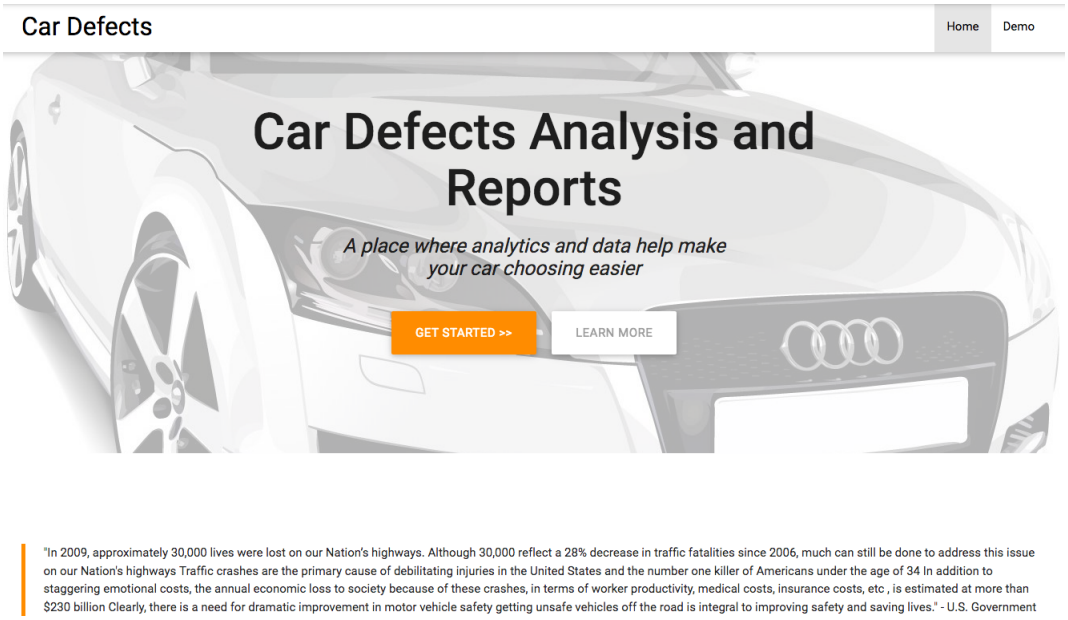


Figure 4.10: Front Page of the Vehicle Defect Analysis Website.

The defects of the top 50 vehicle models (shown in Figure 4.11) which received the most complaints in the NHTSA database are shown in this website. Users can select a model and the corresponding production year, for example "FORD Explorer 2002", "Chevrolet Cobalt 2005", and "TOYOTA Camry 2007". In this website, we take the same product model made in different years as different product models. In this way, we can find the defect difference between these models.

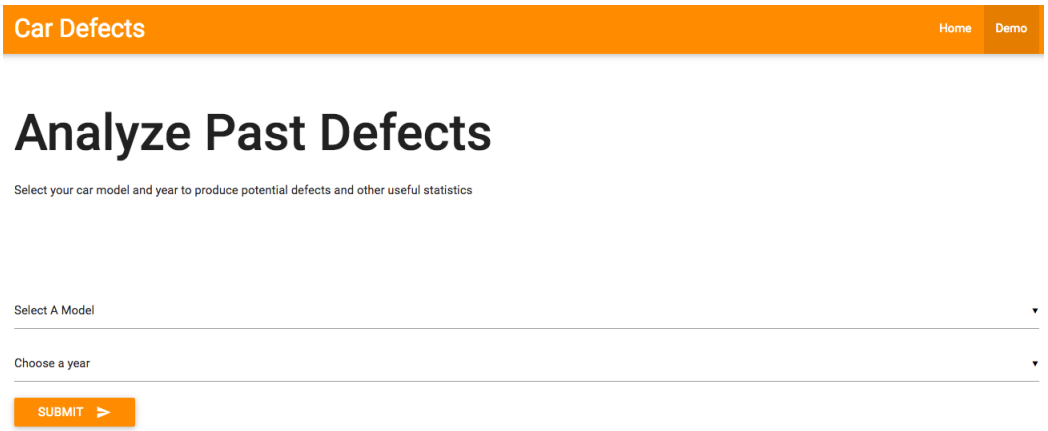


Figure 4.11: Vehicle Model List.

Once we select a model (and a production year), a pie chart (Figure 4.12) shows the top 10 defects of that model. This chart has 10 slices, and each slice represents a type of defect. We use 5 keywords, which are the top 5 words of the component topic of that defect, to represent that defect. The size of the slice is determined by the probability (e.g., 13% in the largest slice) of that defect.

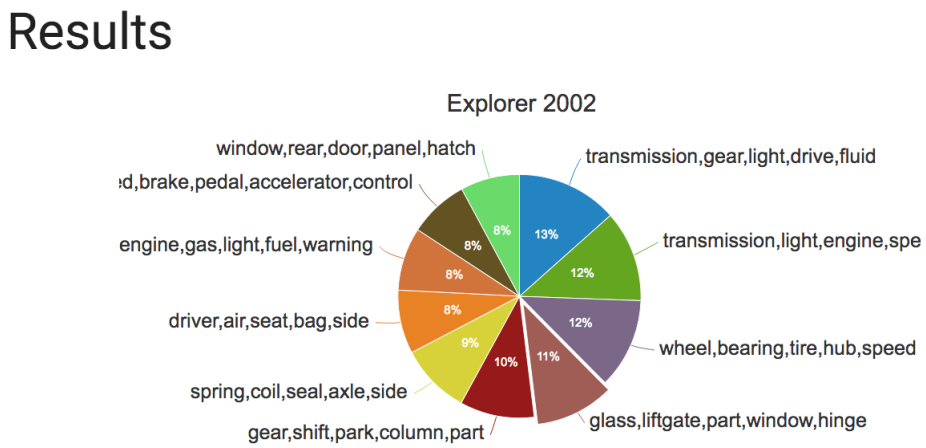


Figure 4.12: Defects of a Vehicle Model.

If the user clicks one slice in Figure 4.12, we can get more information of the corresponding defect, shown in Figures 4.13 and 4.14. For each defect, the symptom and resolution topics are shown as "Word Cloud" (Figure 4.13), in addition to the component topic words. In those Word Clouds, the size of the word depends on its probability in the topic. For example, if we choose the "body and structure" defect of FORD Explorer 2002 in Figure 4.12, we can see the most relevant component words are "glass", "liftgate", "part", "window", etc. The Word Cloud of the symptom topic indicates the door or rear liftgate cracked when this defect

happened. The words in the resolution Word Cloud tell us those customers tried to resolve the window glass problem using their warranty.

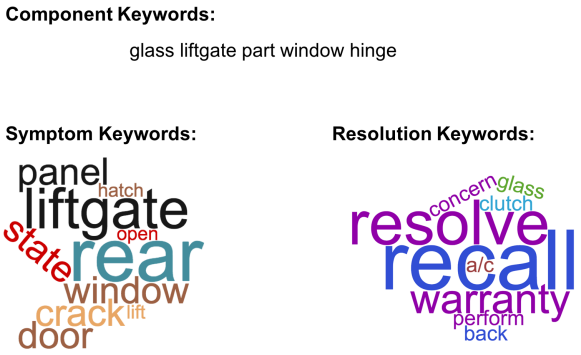


Figure 4.13: Topics of a Defect of FORD Explorer 2002.

Besides the 3 types of topics, the most representative reviews of a defect are shown as well (Figure 4.14). In this example, a few representative reviews related with the "body and structure" are shown. There is some overlap because replicates exist among the NHTSA records.

Representative Complaints

ID	Description	Date
1	THE VEHICLE'S TAILGATE GLASS EXPLODED WHILE THE CONSUMER ATTEMPTED TO CLOSE IT. THE DEALER WAS NOTIFIED. *NLM BROKEN LIFTGATE GLASS AND WIPER MOTOR AND ARM WERE REPLACED. THERE WERE SQUEAKS WHEN GOING OVER BUMPS. SWAY BAR BUSHINGS REPLACED. IGNITI	Sat Apr 05 2003
2	THE VEHICLE'S TAILGATE GLASS EXPLODED WHILE THE CONSUMER ATTEMPTED TO CLOSE IT. THE DEALER WAS NOTIFIED. *NLM BROKEN LIFTGATE GLASS AND WIPER MOTOR AND ARM WERE REPLACED. THERE WERE SQUEAKS WHEN GOING OVER BUMPS. SWAY BAR BUSHINGS REPLACED. IGNITI	Sat Apr 05 2003
3	THE VEHICLE'S TAILGATE GLASS EXPLODED WHILE THE CONSUMER ATTEMPTED TO CLOSE IT. THE DEALER WAS NOTIFIED. *NLM BROKEN LIFTGATE GLASS AND WIPER MOTOR AND ARM WERE REPLACED. THERE WERE SQUEAKS WHEN GOING OVER BUMPS. SWAY BAR BUSHINGS REPLACED. IGNITI	Sat Apr 05 2003
4	DT*: THE CONTACT STATED THE VEHICLE'S LIFT GATE STRUT DISENGAGED, IT FRACTURED THE GLASS AND THE GLASS SHATTERED. THERE WAS A RECALL ON THE NHTSA CAMPAIGN #04V442000, CONCERNING THE LIFT GATE AND ATTACHMENTS. THE RECALL WORK WAS PREFORMED; ALTHOUGH, TH	Sat Feb 11 2006

Figure 4.14: Representative Reviews of a Vehicle Defect.

Chapter 5

Abstractive Summarization of Product Reviews with Hierarchical Encoder-Decoder Neural Network

Along with the emerging of Web 2.0 and mobile computing, people get used to posting thoughts and stories online in short text forms (e.g., product reviews and tweets). Every minute, hundreds of thousands of messages are produced on the Internet, and read by diverse groups of people. Customers want to learn of the strengths, weaknesses, and user experience regarding a product through other people's reviews, especially before making purchasing decisions. Similarly, manufacturers always want to know customers' feedback on their products. However, the vast volume of such User Generated Contents (UGC) leads to a bottleneck when trying to consume them.

In this chapter we discuss research on the problem of generating an abstractive summary of a given collection of product reviews, which is indeed an Abstractive Multiple-Document Summarization (AMDS) task. Assuming a limited number of "topics" (or "stories") are discussed in the collection, the proposed method will output a summary whose words may be "novel" and unseen in the original reviews.

In contrast to Single-Document Summarization (SDS) (Jurafsky, 2000), the Multiple-Document Summarization (MDS) problem has not been thoroughly investigated. Existing methods for multiple-document summarization include the sentence/snippet extraction based method (Jurafsky, 2000; Nguyen et al., 2015) and topic modeling (Blei et al., 2003). However, these methods have weaknesses in terms of flexibility, coherence, semantic/syntactic errors, and summary interpretability. One particular problem with extractive summarization is the summary coherence: as sentences are extracted and concatenated to each other, it is quite

[1] `http://www.dataminingapps.com/2016/02/automated-text-summarization-a-short-overview-of-the-task-and-its-challenges/`

common that there exists no smooth transition between ideas/concepts/topics in different sentences[1]. Inspired by the success of deep neural networks (LeCun et al., 2015) in SDS (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017), we plan to explore their performance on MDS, especially the product review summarization problem.

To date, only very limited work (Liu et al., 2018; Wang and Ling, 2016) has been done regarding abstractive MDS. In both of these two publications, all of the source documents are concatenated into a long sequence, which is then taken as the input of their neural networks. By using the concatenation strategy, they just convert the MDS task into a SDS task indeed. This assumes dependency and order among the source documents, which may not be a valid assumption, and the approach may face the well-known long-term dependency challenge if using RNN to model long sequences (Goodfellow et al., 2016). Such issues may lead to a negative influence on the summary generation. To tackle this problem, we propose methods which use an hierarchical encoder (RNN+FNN, RNN+CNN, or CNN+CNN) to eliminate improper dependencies and avoid problems with modeling a huge sequence. Our hypothesis is that the performance of AMDS can be improved by our novel models which eliminate improper dependencies between documents in the given collection. This hypothesis is built upon the fact that the source data (i.e., review clusters) has an hierarchical structure: a review cluster (or collection) has multiple reviews, and each review has a sequence of tokens.

The contribution of this research includes the following.

- Three Hierarchical Encoder-Decoder Neural Network models which better model an input review collection, and eliminate the dependency and order between its documents. Such dependency used to be improperly modeled by existing neural networks for text summarization.
- A manually labeled dataset for abstractive product review summarization with more than 2,000 ⟨Review Cluster, Summary⟩ instances.
- A dataset created by distant supervision with more than 10,000 ⟨Review Cluster, Summary⟩ records.

In this chapter, Section 5.1 introduces the models for product review summarization, including algorithms for data preprocessing, extractive summarization, as well as neural network models for abstractive summarization. Then, Section 5.2 presents the datasets, evaluation criteria, and baseline methods for our experiments, followed by the quantitative and qualitative evaluation results.

5.1 Approach

To train supervised neural network models for the AMDS task is our ultimate goal, however, we are facing the shortage of a large labeled dataset. This is also an important issue that hin-

ders the development of AMDS research. Two methods have been applied to solve the data shortage problem. First, we manually created a dataset for product review summarization with the help of undergraduate students from the Pamplin College of Business at Virginia Tech. The dataset has review collections of more than 2000 products sold on Amazon.com. A summary is written and checked for each review collection. This labeling process is quite expensive although it can produce high quality summary labels. However, tens of thousands of samples are needed to train deep neural network models. Therefore, we propose another method, a "distant supervision" approach, to produce large but noisy training datasets.

Figure 5.1 shows the distant supervision strategy, which has two main steps. In step 1, we generate text summaries from the given document collection by extracting representative text snippets and cleaning them, and use them as training data for AMDS models. In step 2, we develop several hierarchical Encoder-Decoder neural network models which generate abstractive summaries based on the input text clusters.

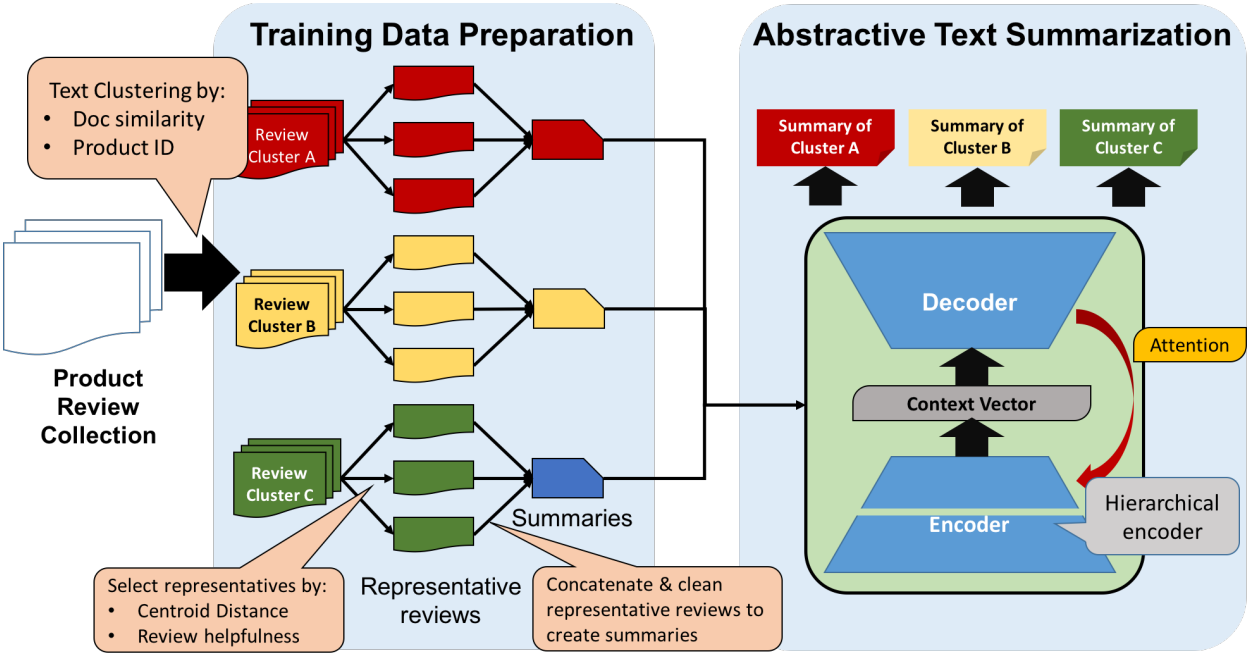


Figure 5.1: A Distant Supervision Approach to Develop Multiple-Document Summarization Models

Given a collection of many short text documents, we first split it into N clusters by clustering methods (e.g., K-Means) or by grouping on product ID. Thus the texts in one cluster are supposed to be about the same topic or the same thing. For each cluster, we choose M representative reviews as candidates for its summary. An important operation is to remove a review from the source collection once it is selected as representative. This can avoid overlap between source and target data, and thus ensure the created dataset is for abstractive rather than extractive summarization. Next, the most important sentences of each representative

review are extracted by extracting the human-written summary of that review, or using unsupervised methods such as TextRank (Mihalcea and Tarau, 2004). The purpose of this step is to identify the key contents of the document, while at the same time avoiding too much detail. Key sentences from multiple reviews are concatenated after removing duplicates. We assume those cleaned key sentences can serve as a fair summary of the cluster, which will be used to train the abstractive summarization models.

For the abstractive summarization step, we develop several neural network models with hierarchical Encoder-Decoder architecture. Since each input to these models is a review cluster with multiple documents, we have a two-level encoder to produce a semantic representation for them. The bottom-level encoder is designed to model each document; based on its output, the top-level encoder will produce a context vector as a representation of the entire review cluster. Afterwards, this vector will be fed into a decoder which generates the summary sequence. The attention (Bahdanau et al., 2014) mechanism might be incorporated into the decoder which will highlight the most relevant documents when generating the next word of the summary sequence.

5.1.1 Distant Supervision

We leverage two distant supervision methods to build document clusters and corresponding summaries.

Preprocessing. For both methods, we first remove all blank reviews. Then we apply the Langdetect Python Package[2] to remove the non-English reviews as well, since they might have a negative effect on our language model.

Method 1: Doc2Vec + K-Means + TextRank. This method is shown in Figure 5.2. Based on our dataset, we train a Doc2Vec model (Le and Mikolov, 2014) with the Gensim Python Package (Řehůřek and Sojka, 2010) and produce a vector for each review. Then, K-Means takes the text features to get text clusters composed of similar texts. The centroid of each cluster is calculated by taking the arithmetic mean. Each text will be assigned to the closest cluster according to document-centroid distance. The reviews that are the closest to the cluster centroid are picked as the summary candidates. To generate the selective summaries, we use the TextRank tool (Mihalcea and Tarau, 2004) to extract the most representative sentences from the reviews as summaries. The length of each summary is between 10 and 50 words. The dataset created using Method 1 is introduced in Section 5.2.

Method 2: Product ID + Helpful Score + Clone Detection. This method is demonstrated by Figure 5.3. It is intuitive to group all the reviews into clusters according to their product ID. Therefore, we sort them by products and select products with more reviews to create the ⟨Text Cluster, Summary⟩ dataset. Afterwards, we take the "helpful" field into

[2] `https://pypi.org/project/langdetect/1.0.6/`

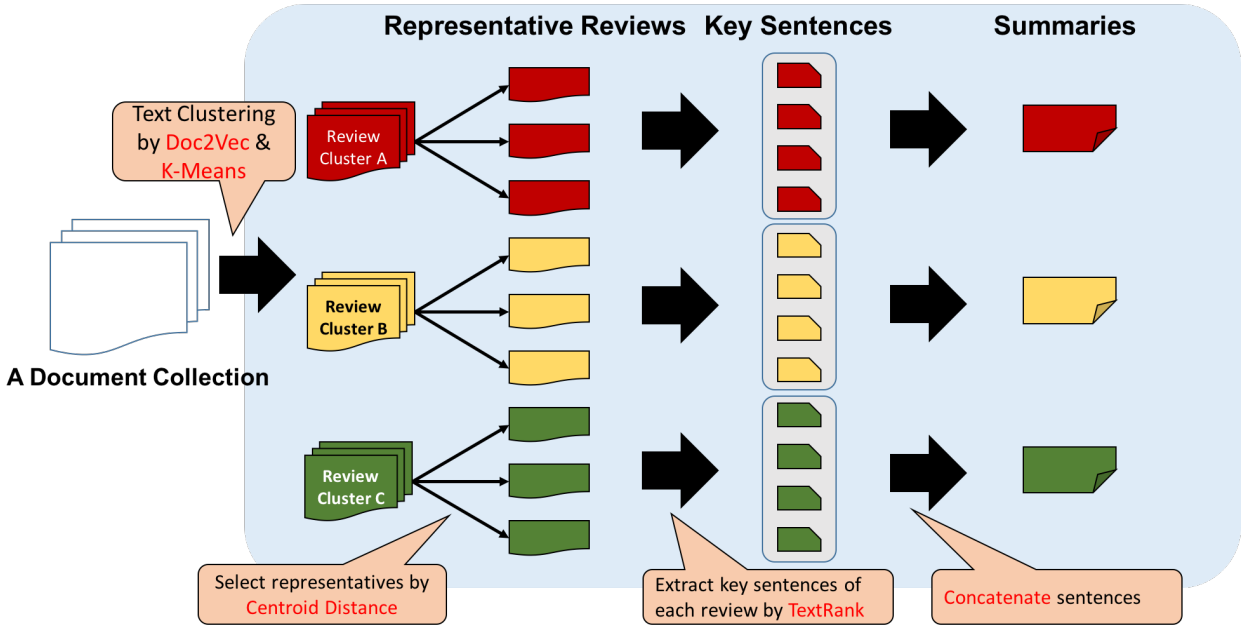


Figure 5.2: Distant Supervision Method 1 to Create Training Data

account. The field format is ⟨X, Y⟩, where X represents the number of "helpful" votes of a given review while Y represents the total number of votes. Based on our assumption, a review with a higher "helpful" score is more likely to be a good representative of customers' opinion on the given product, thus could be considered as a summary. Therefore, we sort the scores in descending order and select the most "helpful" reviews. As most of the Amazon reviews have a short summary (usually between 2 and 20 words) written by the review author, we assume that these review-level summaries can be good "material" to create the cluster-level summary. We remove the short summaries which have less than Min_{sen} words, for example $Min_{sen} = 8$, since we want longer sentences in the abstractive summaries. Then, the remaining short summaries are concatenated to create a cluster-level summary, which can be regarded as an abstractive summary. We say abstractive here because the review-level summaries used to create the cluster-level summary never appear in the body of source reviews. During the concatenation process, we utilize the clone detection method introduced in Liu et al. (2018) defined by Equation (5.1) to filter out the duplicated short summaries, before adding them to the candidate short summary set ($summary - set$ in Equation (5.1). Here, d is a candidate summary for $summary - set$, and a is a summary already existing in $summary - set$. Given a short summary, Equation (5.1) calculates the maximum overlap (recall) it has with the short summaries already put into $summary - set$. The dataset created using this method is introduced in Section 5.2 as well.

$$r(d) = \max_{s \in summary-set} \frac{|unigrams(d) \cap unigrams(s)|}{|unigrams(s)|} \tag{5.1}$$

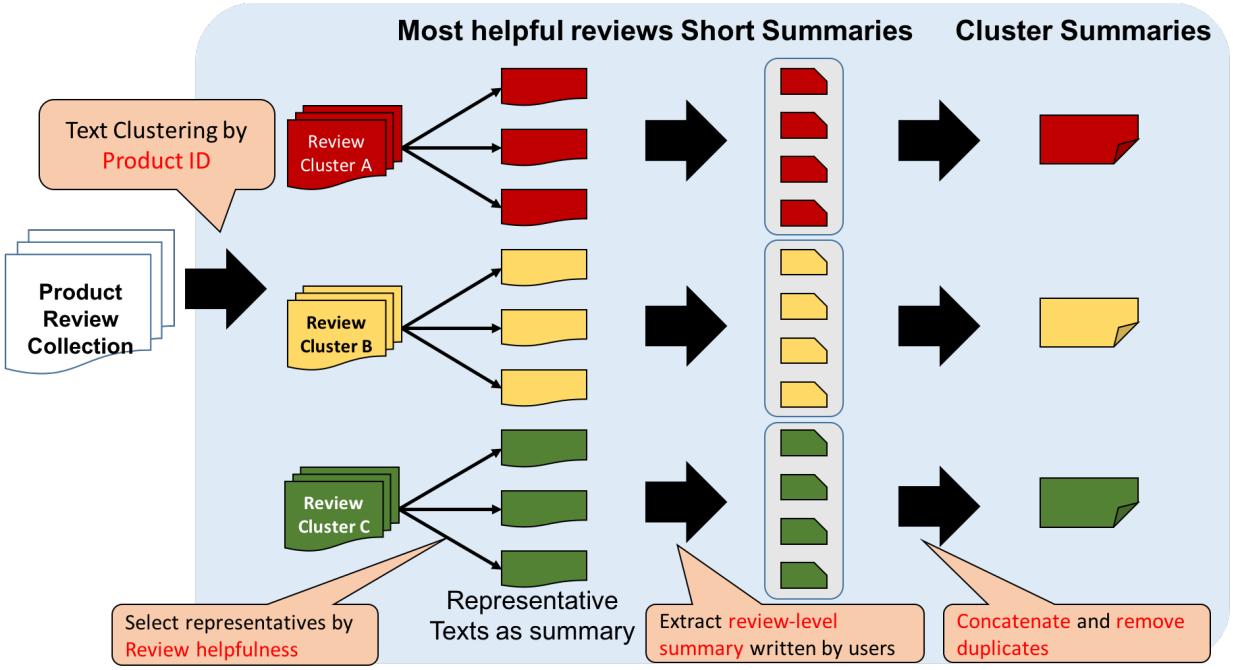


Figure 5.3: Distant Supervision Method 2 to Create Training Data

5.1.2 Abstractive Summarization Models

Our abstractive summary models are three variants of the sequence-to-sequence model introduced by Sutskever et al. (2014), shown in Table 5.1. In general, all of them have a two-level hierarchical encoder structure: a document-level encoder constructed by Recurrent Neural Network (RNN) model (Fausett, 1994), a collection-level encoder built by various neural network components (e.g., FNN and CNN), and a RNN decoder to produce summaries. The difference between the three models occurs in the cluster-level encoder and the decoder. Model 1 uses the simplest architecture, a Feed-forward Neural Network (FNN) to implement the cluster-level encoder. Model 2 replaces the FNN encoder with a more flexible CNN structure. Model 3 has more changes: an attention (Bahdanau et al., 2014) mechanism has been incorporated into the decoder of Model 3, in addition to the CNN cluster-level encoder.

In all of the three models, each document in a collection is encoded by bidirectional Gated Recurrent Units (Cho et al., 2014), an extension of RNN. Before applying RNN, the GloVe pre-trained word embeddings (Pennington et al., 2014) are used to initialize the word embeddings of source and target text, if the words can be found in GloVe. For the rare words not existing in GloVe, we initialize them randomly. For each document j, the hidden states (h_j^F and h_j^B in Equation (5.2), respectively) of the head and the tail of the text sequence are concatenated and go through a linear transformation with weights W_{dc} and bias b_{dc} to get a context vector of document j. Here the weights W_{dc} and bias b_{dc} are parameters which are learned during model training. It's worth noting that all documents are encoded by only

one RNN.

Table 5.1: An Overview of AMDS Models with RNN-based Encoder

Model	Details
1	RNN-FNN Encoder + RNN Decoder
2	RNN-CNN Encoder + RNN Decoder
3	RNN-CNN Encoder + RNN Attention Decoder

$$h_j = W_{dc}[h_j^F, h_j^B] + b_{dc} \tag{5.2}$$

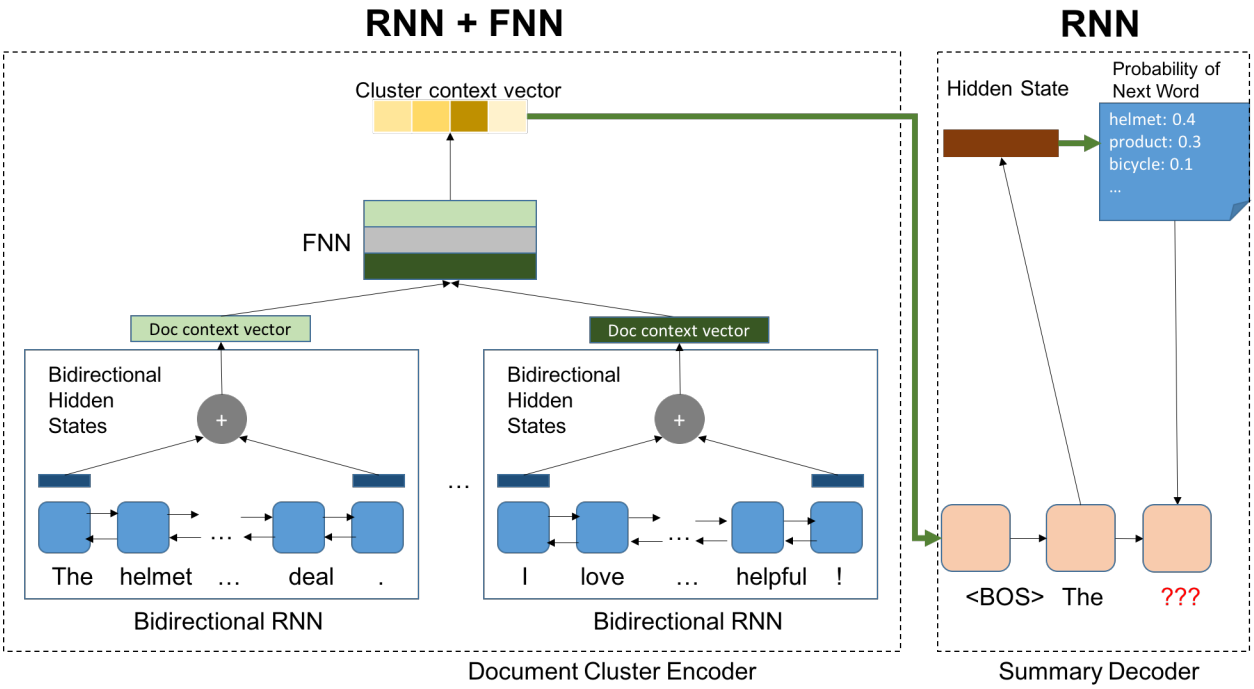


Figure 5.4: A Sequence-to-Sequence Neural Network with a RNN + FNN Encoder

Model 1: RNN-FNN Encoder + RNN Decoder Model. Shown in Figure (5.4), this basic model uses a FNN (Schmidhuber, 2015) to combine the context vectors of documents in the given cluster. Specifically, the context vectors (h_s^C, ..., h_e^C of the cluster documents in Equation (5.3)) are concatenated, then go through a linear transformation (W_{fc} and b_{fc}) layer and an activation ($tanh$) layer. Here, W_{fc} and b_{fc} are parameters to be learned during model training. Afterwards, a context vector is produced as the semantic representation of the document collection, which will be used to initialize the hidden state of the decoder RNN. A unidirectional GRU is used to implement the decoder. At the beginning, a greedy

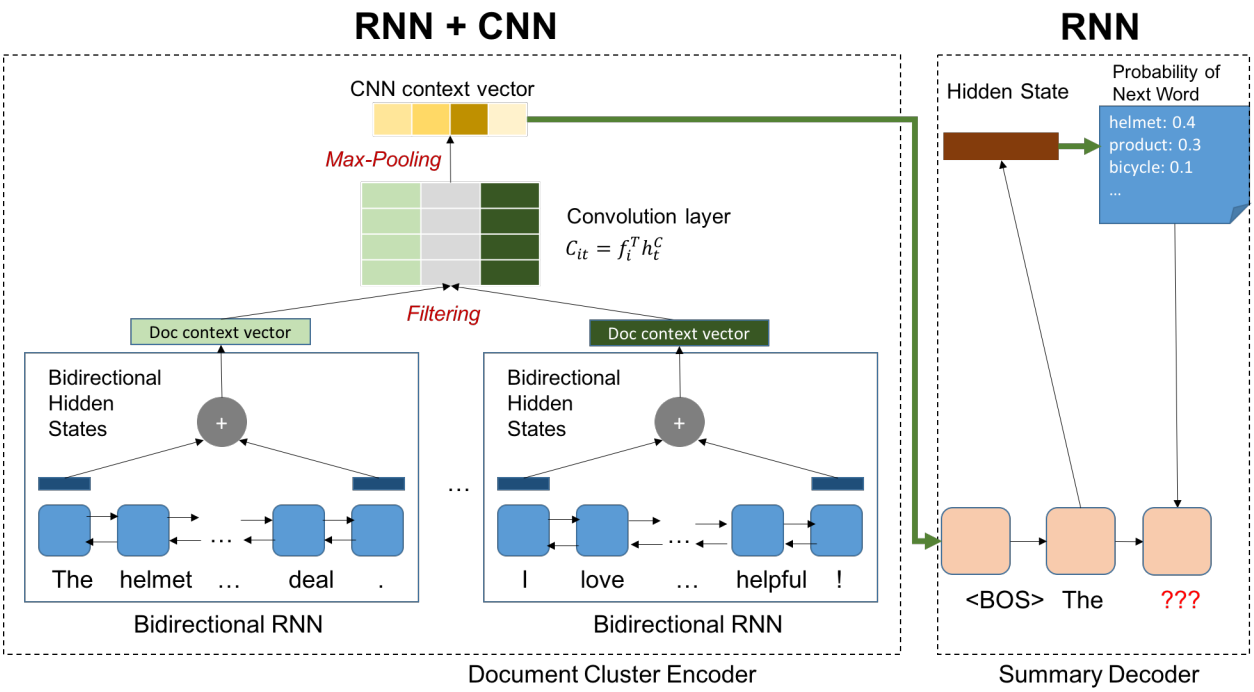


Figure 5.5: A Sequence-to-Sequence Neural Network with a RNN + CNN Encoder

strategy is used during the decoding of the summary, which means we always choose the most probable term when generating the next word. Then it's replaced by beam search (Freitag and Al-Onaizan, 2017), which enlarges the space when searching for a globally optimal solution for the decoding. Basically, beam search maintains a size N beam; the top N words will be used to construct the beam when decoding the next word. All sequences within this beam will be considered. Along the decoding progress, the sequences within a beam will be filtered according to the joint probability of their words. The masked average cross entropy loss defined in Equation (5.4) is used for the model training. Here, $P(w_t)$ is the probability of the tth word in the target sequence. In this model, a heavy burden is put on the cluster context vector, which works as the only method passing the document cluster information. This brings potential risk in the generation of complicated summaries. Another drawback of this model is its weak flexibility, as the dimension of W_{fc} requires a fixed number of documents in a cluster.

$$h_c^F = tanh(W_{fc}[h_s^C, ..., h_e^C] + b_{fc}) \tag{5.3}$$

$$Loss = \frac{1}{T}\sum_{t=1}^{T} -logP(w_t|w_1, ..., w_{t-1}) \tag{5.4}$$

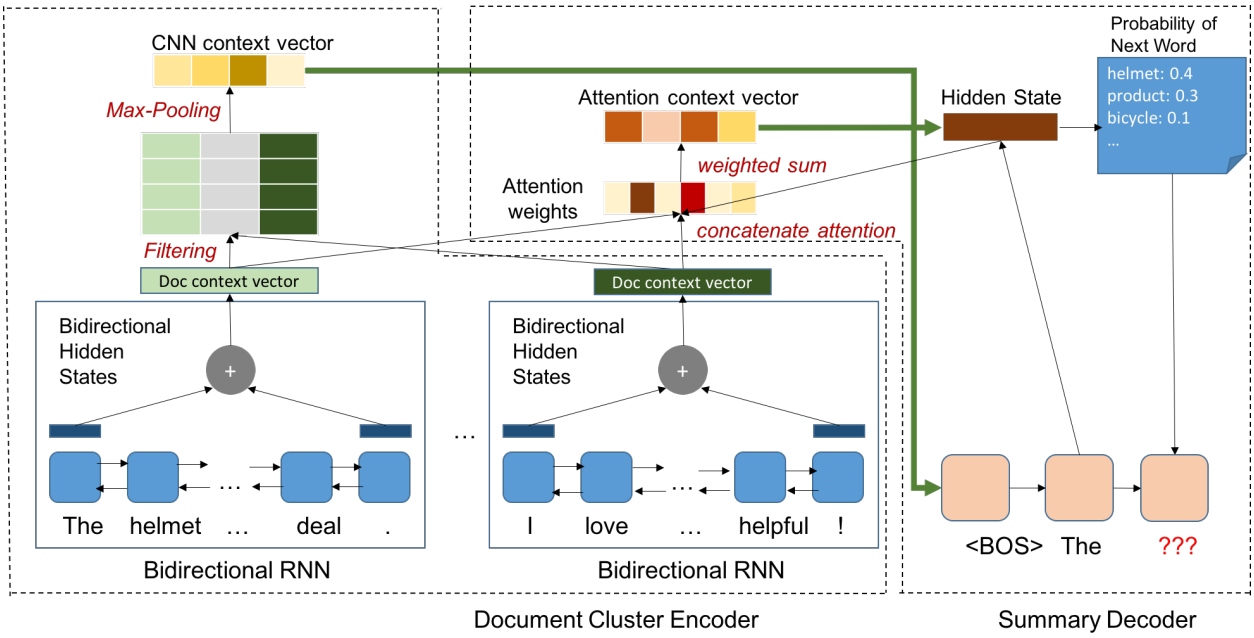


Figure 5.6: A Sequence-to-Sequence Neural Network with a RNN + CNN + Attention Encoder

Model 2: RNN-CNN Encoder + RNN Decoder Model. The design of this model draws inspiration from Wang et al. (2017). As shown in Figure (5.5), a Convolutional Neural Network (CNN) component (LeCun et al., 1998) is used to combine the document context vectors in the given cluster. We apply a convolutional layer with n filters to handle the document context vectors. Following the method of Wang et al. (2017), the size of the filter is set to be the same as the document context vector, which makes the filter scan one document vector each time. This convolutional operation on the vector of document t is defined by Equation (5.5). Afterwards, the batch normalization (Ioffe and Szegedy, 2015), ReLU activation function (Glorot et al., 2011), and max pooling calculations are performed. Finally, a linear layer defined in Equation (5.6) is used to merge the results of n filters, to produce a cluster context vector. Here, P is the vector produced by max pooling. Compared to the first model, which uses FNN as the second encoder layer, this model is more flexible, since it has no limit on the number of documents in a cluster. In Model 2, f_i^T, W_{cc}, and b_{cc} are parameters to be learned.

$$C_{it} = f_i^T \cdot h_t^C \tag{5.5}$$

$$h_c^C = W_{cc}P + b_{cc} \tag{5.6}$$

Model 3: RNN-CNN Encoder + RNN Attention Decoder Model.

In contrast to the previous methods, this model applies an attention mechanism (Bahdanau et al., 2014) to the decoder, which simulates the document-level attention when generating a new word. When humans are doing summarization, they are supposed to pay attention to the salient documents in a cluster. In our model, the hidden state of the decoder and the output of the encoder are incorporated in the computing of attention weights. Both the "dot" scheme defined by Equation (5.8) and the "concat" scheme defined by Equation (5.9) (Luong et al., 2015) have been used in the calculation of the "energy" e_{ij} needed by attention weights. Then, the weight α_{ij} of the jth document when producing the ith word in the summary is calculated according to Equation (5.7). Here, s_{i-1} is the previous hidden state when generating the ith word, while h_j is the output vector of the jth document. In Equation 5.9, W_{en}, W_{cct}, and b are parameters which are learned during model training. By considering these weights, the encoder produces a dynamic cluster-level context vector, and uses it two times: (1) concatenate it with the previous word embedding as the input of the RNN; (2) concatenate it with the RNN output as the input of the term prediction linear layer, which has output dimension equal to the vocabulary size. This two-way incorporation of the context vector will reinforce the influence of the attention context. Finally, the decoder picks the next word based on the result of this linear layer. The detailed explanation of each component is shown in Figure (5.6).

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{D_x} exp(e_{ik})} \tag{5.7}$$

$$e_{ij} = a(s_{i-1}, h_j) = s_{i-1} \cdot h_j \tag{5.8}$$

$$e_{ij} = a(s_{i-1}, h_j) = W_{en} \cdot (W_{cct}[s_{i-1}, h_j] + b) \tag{5.9}$$

5.2 Evaluation

5.2.1 Datasets

Focusing on the short text collections, we choose the *Amazon Product Data*[3] introduced in He and McAuley (2016) and McAuley et al. (2015b) as the main data source in the project.

[3] `http://jmcauley.ucsd.edu/data/amazon/links.html`

Shown in Table 5.2, three datasets have been created in total: one with human-written summaries, and two with the distant supervision methods mentioned in Section 5.1.1.

For Dataset 1, we first select 200 products with the largest number of reviews from each of the 24 categories. For each of the $4,800$ products, we sort their reviews by the "helpful" score to get the top 50 reviews. In the Spring and Summer of 2018, 78 students from the Pamplin College of Business at Virginia Tech were asked to write an opinion summary after reading the 50 reviews of each product. They consider the following opinion aspects when writing summaries: general statement, product strength and weakness, quality, appearance, important user experience, and price. The quality of the summary was manually checked by the researchers after submission. Eventually, we obtained $2,221$ ⟨Review Collection, Summary⟩ pairs, which can serve as a dataset for AMDS model evaluation. This dataset will be extended in the future.

The distant supervision method 2 introduced in Section 5.1.1 is used during the construction of Dataset 2. In particular, we sort all the reviews by product and select products with more reviews to create this dataset. Afterwards, we sort the scores by the "helpful score" in descending order and select the most helpful reviews. Most of the Amazon reviews have a short summary written by the review author, which can be good material to create the cluster-level summary. We select the short summaries with at least Min_{sen} words (e.g., $Min_{sen} = 8$), and remove duplicates with the clone detection method introduced in Liu et al. (2018). Then, the remaining short summaries are concatenated to create a summary of the review collection. Since the short summaries used to create a cluster-level summary never occur in the source reviews, the summaries produced for the clusters can be regarded as abstractive summaries.

For Dataset 3, we select the reviews on "Sports and Outdoors" products to create the dataset. Based on the Doc2Vec representation, K-Means algorithm, and TextRank method, there are a total of 18130 review clusters, and each cluster contains 50 reviews and 10 summaries. We split them into training, validation, and testing sets. The number of clusters in each set is 10870, 3630, and 3630, respectively. The percentages are 60, 20, 20, respectively. The description of this dataset is shown in Column 4 of Table 5.2.

Dataset 1, the manually created dataset, is not big enough for neural network training, and the summaries in Dataset 3 have a great deal of noise. Thus, we take Dataset 2 as the major training dataset during our experiments, because the summaries in this dataset have higher quality and large volume. The prediction results on Datasets 1 and 2 are reported in Section 5.2.5.

5.2.2 Baseline Methods

As introduced previously, there has been very limited research (Liu et al., 2018; Wang and Ling, 2016) on AMDS using deep neural networks. As the implementation and datasets of

Table 5.2: An Overview of AMDS Datasets

Dataset		**Dataset 1**	**Dataset 2**	**Dataset 3**
Clusters	**Total**	2,221	10,084	18,130
	Training	1,776 (80%)	8,067 (80%)	10,870 (60%)
	Validation	222 (10%)	1,009 (10%)	3,630 (20%)
	Testing	223 (10%)	1,009 (10%)	3,630 (20%)
Each Cluster	**# of Reviews**	50	50	50
	Avg Len of Reviews	119.72	84.45	108.66
	Avg Len of Summaries	70.62	38.25	22.08

Liu et al. (2018) have not been released yet, we take the model introduced by Wang and Ling (2016) and Textsum[4], a popular single document summarization model, as baseline methods and adapt them to our environment.

- The first model is the attention-based sequence-to-sequence model (Bahdanau et al., 2014), shown in Figure (5.7), which was proposed for machine translation originally. But different variants have been developed for other tasks including text summarization. In our experiments, we use the Textsum Python Project implemented by Google for this model.
- The other is the Opinion & Argument Abstract Generation (OAAG) Model (Wang and Ling, 2016), shown in Figure (5.8). OAAG is one of the state-of-the-art models for the AMDS task.

To use the Textsum model, the reviews in each review collection are concatenated to form a very long single document, which is taken as the source document to be summarized.

5.2.3 Evaluation Criteria

In this research, we use ROUGE (Lin, 2004) as the primary metric. The calculation of $ROUGE-N\ Recall$ is defined in Equation (5.10). Here, n means the number of tokens in n-gram ($gram_n$), while $Count_{match}(gram_n)$ denotes the maximum count of n-grams in a predicted summary which match the text in the reference summaries. The calculation of $ROUGE-N\ Precision$, which is also called BLEU-n (Papineni et al., 2002), is defined in Equation (5.11). The difference between the equations of recall and precision is the denominator part. The most widely used $ROUGE-N$ metrics are $ROUGE-1$ and $ROUGE-2$, which stand for the matching level of prediction and reference summaries with regard to unigrams and bigrams, respectively. The PyRouge Python Package[5] has been used

[4] `https://github.com/tensorflow/models/tree/master/research/textsum`

[5] `https://github.com/andersjo/pyrouge`

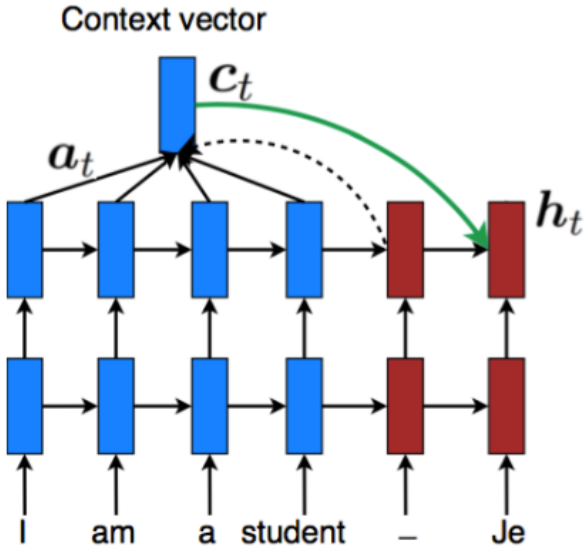


Figure 5.7: Bahdanau Attention Model for Machine Translation

to measure ROUGE in our experiments.

Other metrics such as CIDEr (Vedantam et al., 2015), and METEOR (Denkowski and Lavie, 2011) will be measured in future work.

$$ROUGE-N\ Recall = \frac{\sum_{S \in Reference-Summaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in Reference-Summaries} \sum_{gram_n \in S} Count(gram_n)} \tag{5.10}$$

$$ROUGE-N\ Precision = \frac{\sum_{S \in Candidate-Summaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in Candidate-Summaries} \sum_{gram_n \in S} Count(gram_n)} \tag{5.11}$$

5.2.4 Evaluation Settings

The settings of the models are shown in Table (5.3). In particular, the Adam optimization algorithm (Kingma and Ba, 2014) is used for model training. The "early stopping" scheme is used. We evaluate the average loss on the validation set after each epoch. The model training will stop earlier once the average loss on the validation set begins to increase for 5 epochs. The models saved at the lowest validation loss point are used to do prediction on the test set. Most of the models are trained on the Huckleberry and Cascades clusters of the Advanced Research Computing (ARC) Center of Virginia Tech. The hardware specification of Huckleberry and Cascades is:

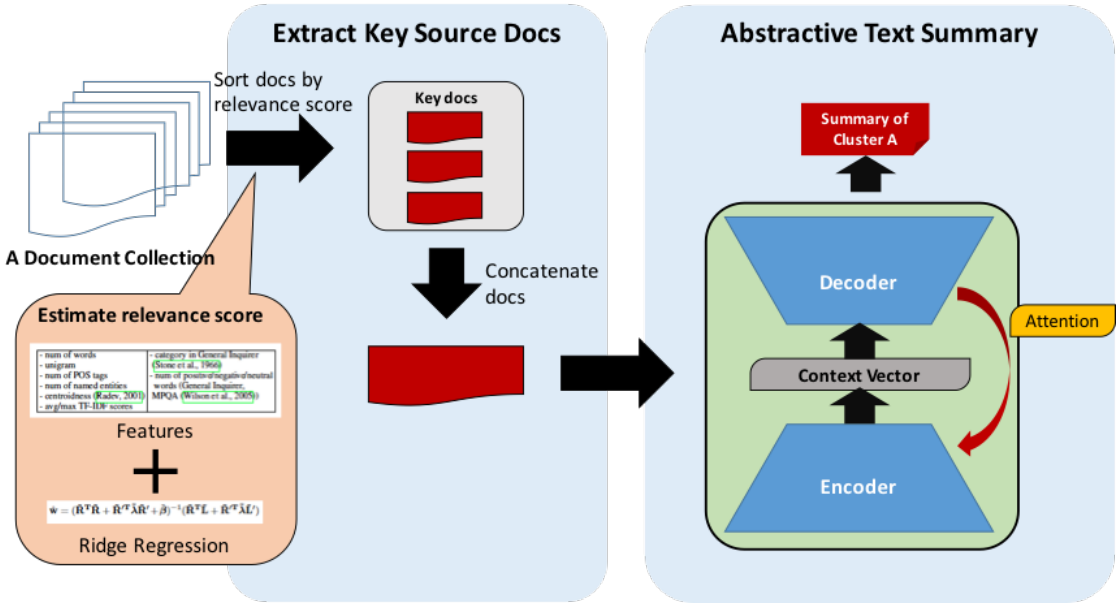


Figure 5.8: Opinion & Argument Abstract Generation Model, adapted from Wang and Ling (2016)

- **Huckleberry**: 14 nodes; each node has 2 IBM Power8 3.26 GHz CPUs, 256 GB memory, and 4 NVIDIA Tesla P100 GPUs with 16 GB of memory.
- **Cascades**: 39 V100 nodes; each node has 2 Intel Skylake Xeon Gold 3 GHz CPUs, 384 GB memory, and 2 NVIDIA Tesla V100 GPUs with 16 GB of memory.

Table 5.3: Hyper Parameters of Models 1-3

Settings	Values
Epochs	50
Batch size	40
Optimization	Adam
Learning rate	0.0001
Docs per cluster	50
Max source length	200
Max target length	220
Encoder CNN/RNN layers	1
Decoder RNN layers	1

5.2.5 Evaluation Results

Both quantitative and qualitative evaluation has been conducted in this research. In quantitative evaluation, we compare the ROUGE metrics of different models, including the baseline methods; sample summaries generated by proposed models are presented in the qualitative evaluation. For both types of evaluation, the results on Dataset 2 (see Table 5.2) are reported.

Quantitative Evaluation Results

The ROUGE metrics of various models on Dataset 2 are shown in Tables 5.4 and 5.5. We can see the proposed models achieve an obvious advantage over the baseline methods on most of the ROUGE metrics, except the precision of ROUGE-1 and ROUGE-2. This suggests that the models with hierarchical encoders apparently outperform models with a single-level encoder in the multiple documents summarization task.

One possible reason can be found in Table 5.6, which presents the average number of words and average unique words in the generated summaries. The number of unique words in the summaries generated by our models was much more than the number generated by the baseline models. Textsum produces very short summaries, although it has applied beam searching. There is little repetition in its summaries, looking at the small difference between the average length and the average number of unique words. In contrast, repetition may frequently be seen in OAAG's summaries; we see a large average length together with a small average number of unique words. Thus, we can infer the relevant information volume in the summaries generated by OAAG is quite small, although they match the ground truth better. This can explain why OAAG achieves a high ROUGE-1 precision score, but yields a poor result in terms of recall.

Among our models, Model 3b (with "concat" attention) produces the best ROUGE-1 and ROUGE-2 F1. This proves the concatenation attention mechanism (even only with document-level attention) is helpful in improving summary quality. Model 2 has a slight advantage over Model 1, which shows the CNN network works better than FNN when modeling cluster context. Regarding the recall score of ROUGE-1, our Model 1 with FNN cluster encoder achieves the best result. In contrast to the baseline models, our models show more balanced performance in terms of precision and recall.

The ROUGE scores in Table 5.4, including the scores of the Textsum baseline method, are lower than those on other datasets reported in the related works (Nallapati et al., 2016). For example, a model similar to Textsum has a 0.3133 ROUGE-1 F1 on the CNN/Daily Mail dataset (Nallapati et al., 2016). There are two main reasons for that:

- The volume of the dataset. Neural network models have a large number of parameters,

thus require a huge amount of data for training. However, the current Dataset 2 has only 8,067 ⟨Source, Summary⟩ pairs in its training data folder. In comparison, the CNN/Daily Mail dataset has far more instances (287,226 pairs for training, 13,368 for validation, and 11,490 for testing), although each instance in our dataset has much more text.

- The summaries in the CNN/Daily Mail dataset are written by humans, which means higher quality. However, the summaries in our Dataset 2 are produced based on distant supervision, as a low-cost solution to the lack of training data.

Therefore, in future work we will improve the quantity and quality of our dataset through extension of the automatically generated datasets and the manually created dataset.

Table 5.4: ROUGE-1 Metrics on Dataset 2

Category	**Model**			**ROUGE-1**		
	Name	**Encoder**	**Decoder**	**P**	**R**	**F1**
Baseline Model	Textsum	RNN	RNN + Concat Attention	**0.3368**	0.1172	0.1634
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	0.2939	0.1400	0.1765
Our Models	Model 1	RNN + FNN	RNN	0.2492	**0.2149**	0.2198
	Model 2	RNN + CNN	RNN	0.2839	0.2021	0.2254
	Model 3a	RNN + CNN	RNN + Dot Attention	0.2372	0.2077	0.2109
	Model 3b	RNN + CNN	RNN + Concat Attention	0.2926	0.2124	**0.2344**

Table 5.5: ROUGE-2 Metrics on Dataset 2

Category	**Model**			**ROUGE-2**		
	Name	**Encoder**	**Decoder**	**P**	**R**	**F1**
Baseline Model	Textsum	RNN	RNN + Concat Attention	**0.0716**	0.0228	0.0328
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	0.0334	0.0166	0.0206
Our Models	Model 1	RNN + FNN	RNN	0.0447	0.0397	0.0399
	Model 2	RNN + CNN	RNN	0.0530	0.0386	0.0424
	Model 3a	RNN + CNN	RNN + Dot Attention	0.0460	0.0424	0.0418
	Model 3b	RNN + CNN	RNN + Concat Attention	0.0592	**0.0442**	**0.0481**

In order to show our models have better performance on the human-labeled datasets, we applied the models trained on Dataset 2 to predict the summaries of Dataset 1. The ROUGE-1 and ROUGE-2 scores are demonstrated in Tables 5.7 and 5.8. We present the summary length statistics in Table 5.9. From those tables, we again can see our models outperform the baseline methods in terms of recall and F1. This time, our Model 2 has the best recall and F1 scores, while the baseline models have better precision. This result indicates our models have better review summarization performance than the baselines when evaluated with the human-labeled datasets.

Table 5.6: Summary Statistics on Dataset 2

Category	**Model**			**Summary Length**	
	Name	**Encoder**	**Decoder**	**Average length**	**Unique words**
Baseline Model	Textsum	RNN	RNN + Concat Attention	13.42	11.71
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	29.29	10.48
Our Models	Model 1	RNN + FNN	RNN	31.83	23.47
	Model 2	RNN + CNN	RNN	26.18	15.12
	Model 3a	RNN + CNN	RNN + Dot Attention	31.48	17.28
	Model 3b	RNN + CNN	RNN + Concat Attention	25.64	16.42

Table 5.7: ROUGE-1 Metrics on Dataset 1

Category	**Model**			**ROUGE-1**		
	Name	**Encoder**	**Decoder**	**P**	**R**	**F1**
Baseline Model	Textsum	RNN	RNN + Concat Attention	**0.4542**	0.0584	0.1009
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	0.3680	0.0770	0.1231
Our Models	Model 1	RNN + FNN	RNN	0.2193	0.1761	0.1932
	Model 2	RNN + CNN	RNN	0.2487	**0.1941**	**0.2144**
	Model 3a	RNN + CNN	RNN + Dot Attention	0.2489	0.1740	0.1931
	Model 3b	RNN + CNN	RNN + Concat Attention	0.2447	0.1936	0.2136

Qualitative Evaluation Results

To complement the quantitative evaluation, we discuss a qualitative evaluation in this section, which has been done on Dataset 2.

Table 5.10 shows an example of the ⟨Review Cluster, Summary⟩ pair in the test set of Dataset 2. The reviews are about an album named "The dark side of the moon" by Pink Floyd. The majority of the customers gave high ratings to this album, except for a few exceptions, which can be seen in both the reviews and the reference summary created. We can observe positive comments like "still awesome today", "great album", "one of the greatest albums of all time", and "brilliant" in the reference summary. The only negative comment that occurs in the reference summary is "bad packaging".

Correspondingly, the summaries generated by different models are demonstrated in Table 5.11.

The summaries generated by the baseline methods, Textsum and OAAG, are shorter and involve more errors. For example, Textsum produces the duplicate text "of the year". The summary generated by OAAG has a semantic error, i.e., "best born album i've ever made",

Table 5.8: ROUGE-2 Metrics on Dataset 1

Category	**Model**			**ROUGE-2**		
	Name	**Encoder**	**Decoder**	**P**	**R**	**F1**
Baseline Model	Textsum	RNN	RNN + Concat Attention	**0.0974**	0.0095	0.0171
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	0.0348	0.0072	0.0115
Our Models	Model 1	RNN + FNN	RNN	0.0299	0.0240	0.0264
	Model 2	RNN + CNN	RNN	0.0409	**0.0318**	**0.0354**
	Model 3a	RNN + CNN	RNN + Dot Attention	0.0325	0.0228	0.0253
	Model 3b	RNN + CNN	RNN + Concat Attention	0.0391	0.0307	0.0339

Table 5.9: Summary Statistics on Dataset 1

Category	**Model**			**Summary Length**	
	Name	**Encoder**	**Decoder**	**Average length**	**Unique words**
Baseline Model	Textsum	RNN	RNN + Concat Attention	10.51	9.57
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	40.77	10.52
Our Models	Model 1	RNN + FNN	RNN	63.22	23.13
	Model 2	RNN + CNN	RNN	61.37	22.84
	Model 3a	RNN + CNN	RNN + Dot Attention	62.24	24.65
	Model 3b	RNN + CNN	RNN + Concat Attention	62.06	22.46

which should not happen because the customers are evaluating an album published by other people. Also, there are some grammar errors at the end of the summary.

In contrast, the summaries produced by our models have greater length and fewer errors, although they are not perfect. Most of the summaries generated by Model 1 through Model 3b can be regarded as a decent paraphrase of the reference summary. In general, these summaries are mostly composed by positive comments like "the best album of all time", "best album i've ever heard", and "great album for the money and a great price". Among all our models, the summary produced by Model 3b seems to cover more topics. It not only states the general comments like "the best album of all time", but also discusses the price of the product.

Another interesting thing happens in the summary of Model 2. It has comments contradicting each other: "but not the best albums of all time" and "a good album of the best albums ever". The appearance of such contradictory contents is because different customers hold different thoughts on the product and they write reviews with diverse polarity. Since this diversity occurs in the source reviews, contradicting sentences or clauses may be generated in the summary.

Compared to human-written summaries, the generated summary still has a weakness in

terms of coherence. Humans may write "Most of the customers consider this as the best album ever" and "only a small number of people think this is not the best album of all time", if we are asked to write the review. But summaries produced by the models are less coherent, just like the contradiction example in Model 2. This is partly caused by the quality of the summary labels, which are created using distant supervision, and thus are less coherent than human-written summaries. Another known issue with the predicted summaries is the shortage of details. The produced summaries seem to be more general than human-written summaries. For example, all of the models have predicted general comments such as "the best album", but haven't caught some details like "bad packaging". One possible reason for the generality is the lack of model training with large datasets. During the training of our models, the models may learn the common parts of those summaries first, which is usually quite general. For details which occur less often in the source reviews, the model may ignore them. This will be investigated in future work.

Table 5.10: An Example of ⟨Review Cluster, Summary⟩ Pair

No.	**Sample reviews**
1	were it not for this dark side of the moon's (dsom) commercial success , i'd probably having given it five stars ... after all , it is pink floyd. but , dsom is as much a reflection of alan parson's production effort ...
2	this release is a wonderful treat for all floyd fans as well as the remastering of the album sounds even better than before , as for bonus material of demos , live , and visuals dj redbeard over-hyped the quantity of these items that ...
...	... ...
49	in my opinion , this is one album that everyone should own . my mom is a huge pink floyd fan , so i grew up listening to this album more times than can be counted . most concept albums are sort of a chore to sit through and the message is so confusing ...
50	this pressing is horrible . it is very noisy . do not confuse this with the 2011 version , which by all accounts sounds great .
No.	**Reference Summary**
1	dark side of the moon pink floyd immersion box set . best album from my youth and still awesome today . great album , bad packaging for the discs . one of the greatest albums of all time ; brilliant !

Time Performance on Model Training

The time performance on model training is shown in Table 5.12. For the training of all the neural network models, only one GPU (on one node) is used, no matter P100 or V100. We can see the Textsum baseline used much more time than the others. The difference between the other models are much smaller. The potential reasons might include:

Table 5.11: Sample Summaries Generated by Different Models

Category	**Model**	**Generated Summary**
Baseline models	Textsum	best album of the year of the year .
	OAAG	best born album i've ever made ! . ! . of pop .
Our models	Model 1	the best album of all time , but still a great album . one of the best albums of the decade ! ! ! the best album i've heard in years .
	Model 2	good , but not the best albums of all time . a good album of the best albums ever . best album i've ever heard !
	Model 3a	one of the best albums i've heard in the best albums . the best album ever ever ever heard in a long time .
	Model 3b	one of my new favorite albums of all time . the best album i have ever owned in the world ! great album for the money and a great price .

- First, the RNN Encoder and Decoder runs quite slowly on very long text sequences (e.g., 2000 words in each cluster). This long time cost can also be found on the OAAG baseline although its text sequence is shorter than Textsum due to the document sampling strategy.
- Second, we use TensorFlow V1.4, an early version, to implement this model.

In fact, Model 3b runs faster than we expected, even faster than Model 3a, whose architecture is simpler because the calculation of Dot Attention is less expensive than Concat Attention. One possible reason is the hardware difference (Nvidia Tesla P100 vs. V100).

Table 5.12: Time Used for 50 Epochs Model Training on Dataset 2

Category	**Model**			**Environment & Timing**		
	Name	**Encoder**	**Decoder**	**GPU**	**Platform**	**Time(h)**
Baseline Model	Textsum	RNN	RNN + Concat Attention	Nvidia Tesla P100	TensorFlow 1.4	139.93
	OAAG	RNN + Doc Sampling	RNN + Concat Attention	Nvidia Tesla V100	PyTorch 0.3	19.98
Our Models	Model 1	RNN + FNN	RNN	Nvidia Tesla P100	PyTorch 0.3	17.66
	Model 2	RNN + CNN	RNN	Nvidia Tesla V100	PyTorch 0.3	13.02
	Model 3a	RNN + CNN	RNN + Dot Attention	Nvidia Tesla V100	PyTorch 0.3	19.62
	Model 3b	RNN + CNN	RNN + Concat Attention	Nvidia Tesla P100	PyTorch 0.3	12.34

Chapter 6

Contributions and Future Work

6.1 Contributions

This research introduces novel topic modeling and neural network techniques for product defect discovery from user reviews.

As the main contributions of this study, novel PLSA, LDA, and Hierarchical Encoder-Decoder neural network models are proposed, respectively, to identify the most complained about defects from user reviews, capture the key defect entities, and summarize the reviews. All of these methods take product defect discovery as the primary goal, but their functionality has small differences. PDM takes reviews of various product models as input, and produces joint topics on multiple facets such as Model, Component, and Symptom. People may use PDM to identify the most criticized defects in product reviews posted in online stores, such as Amazon and BestBuy. In contrast, PDLDA focuses on the reviews of one product model at a time, and generates interdependent topics on Component, Symptom, and Resolution. PDLDA might be used to analyze user discussions in product forums, or repair/maintenance records of mechanics. Regardless of the difference in functionality, an entity extraction step is needed by both topic models, which extracts necessary entities from reviews using techniques such as frequent item set mining, text classification, and a lexicon. Afterwards, extracted entities are fed into topic models, which will infer topic assignment on these entities and produce various types of topics. These topic models will be very helpful if people want to learn the resolution to an issue. In contrast, the Hierarchical Encoder-Decoder neural network models generate abstractive summaries in the form of natural language sentences, whose output has better readability and is easier to interpret.

The details of the contributions are shown below:

- A Probabilistic Defect Model (PDM) for extracting product quality issues from unstructured UGC. PDM captures the product models, flawed components, and symp-

toms related with each defect.

- A probabilistic method to find the most related reviews of a defect using PDM.
- Three manually labeled datasets created for the product review clustering evaluation. Each of them has more than 1,000 review records, which are based on the NHTSA vehicle complaint database. Six larger NHTSA and four Amazon review datasets have been created for qualitative evaluation.
- A Product Defect Latent Dirichlet Allocation (PDLDA) model for identifying product defects and corresponding resolutions from user reviews. PDLDA is able to extract defective components, symptoms, and resolutions related with each product issue. The inter-dependency of those entities is also modeled with PDLDA.
- The elimination of the dependence on word pairs of existing multiple-facet LDA models by PDLDA. Existing LDA models (Moghaddam and Ester, 2011; Wang and Ester, 2014; Xue et al., 2015) for aspect summarization usually require word pairs (e.g., an aspect word and a rating word) as input. In contrast, PDLDA accepts user reviews which have different numbers of component, symptom, and resolution words.
- Three datasets created for the product review entity clustering evaluation. One for FORD vehicle product (1,941 records), one for APPLE MacBook product (413 records), and one for the patient.info healthcare forum (43,928 records). Among them, the first two datasets are manually labeled, while the reviews in the patient.info dataset are labeled by their subforum name. Three similar but larger datasets have been created for qualitative evaluation.
- Three Hierarchical Encoder-Decoder Neural Network models which eliminate the dependency and order between documents in a review collection. Such dependency used to be improperly modeled by existing neural networks for text summarization.
- A manually labeled dataset for abstractive product review summarization with more than 2,500 instances. Also developed is a dataset created by distant supervision with more than 10,000 records.

6.2 Publications

A number of papers have been published during my Ph.D. program. A selected set of publications are shown in the list below.

- Prashant Chandrasekar, **Xuan Zhang**, Saurabh Chakravarty, Arijit Ray, John Krulick, Alla Rozovskaya, "The Virginia Tech System at CoNLL-2016 Shared Task on Shallow Discourse Parsing", Proceedings of the 54th Annual Meeting of the Association for

Computational Linguistics (ACL), pages 115 - 121, 2016, Berlin, Germany. The first two authors have equal contribution. (Chandrasekar et al., 2016).

- **Xuan Zhang**, Zhilei Qiao, Lijie Tang, Weiguo Fan, Edward Fox, Alan G. Wang, "Identifying Product Defects from User Complaints: A Probabilistic Defect Model", Proceedings of 22nd Americas Conference on Information Systems (AMCIS), 2016, San Diego, CA, USA. (Zhang et al., 2016).
- **Xuan Zhang**, Mi Zhou, Weiguo Fan, Alan G. Wang, "Automatic Recognition of Adverse Events in News: a Data Mining Approach", Workshop on Information Technologies and Systems (WITS), December 2014, Auckland, New Zealand. (Zhang et al., 2014).
- **Xuan Zhang**, Shuo Niu, Da Zhang, G. Alan Wang, Weiguo Fan, "Predicting Vehicle Recalls with User-Generated Contents: A Text Mining Approach", Proceedings of Pacific Asia Workshop on Intelligence and Security Informatics (PAISI) 2015, pp.41-50, May 2015, Ho Chi Minh City, Vietnam. (Zhang et al., 2015).
- Zhilei Qiao, **Xuan Zhang**, Mi Zhou, Alan G. Wang, Weiguo Fan, "A Domain Oriented LDA Model for Mining Product Defects from Online Customer Reviews", Proceedings of the 50th Hawaii International Conference on System Sciences (HICCS), 2017. (Qiao et al., 2017).
- Tarek Kanan, **Xuan Zhang**, Mohamed Magdy, Edward Fox, "Big Data Text Summarization for Events: a Problem Based Learning Course", Short Paper in Joint Conference on Digital Library (JCDL) 2015, June 2015, Knoxville, Tennessee, USA. (Kanan et al., 2015).

6.3 Future Work

We will continue the work on the three research projects, and report results, as follows.

- For each of the three projects, we plan to release the related datasets and code upon the acceptance of the corresponding papers.
- For the PDLDA project, additional quantitative evaluation such as Precision@N will be performed.
- In terms of the product review summarization research, we'll extend both the human-labeled and the distant supervision datasets. Novel neural network architectures, such as self-attention (Lin et al., 2017) and Transformer (Vaswani et al., 2017), will be adapted to further improve the model performance.

References

Abrahams, A. S., Fan, W., Wang, G. A., Zhang, Z. J., and Jiao, J. (2015). An integrated text analytic framework for product defect discovery. *Production and Operations Management*, 24(6):975–990.

Abrahams, A. S., Jiao, J., Fan, W., Wang, G. A., and Zhang, Z. (2013). What's buzzing in the blizzard of buzz? Automotive component isolation in social media postings. *Decision Support Systems*, 55(4):871–882.

Abrahams, A. S., Jiao, J., Wang, G. A., and Fan, W. (2012). Vehicle defect discovery from social media. *Decision Support Systems*, 54(1):87–97.

August, T. and Niculescu, M. F. (2013). The influence of software process maturity and customer error reporting on software release and pricing. *Management Science*, 59(12):2702–2726.

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

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Buell, R. W., Campbell, D., and Frei, F. X. (2016). How do customers respond to increased service quality competition? *Manufacturing & Service Operations Management*, 18(4):585–607.

Chandrasekar, P., Zhang, X., Chakravarty, S., Ray, A., Krulick, J., and Rozovskaya, A. (2016). The virginia tech system at conll-2016 shared task on shallow discourse parsing. *Proceedings of the CoNLL-16 shared task*, pages 115–121.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Chen, Y. and Xie, J. (2008). Online consumer review: Word-of-mouth as a new element of marketing communication mix. *Management science*, 54(3):477–491.

Chen, Y.-C. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Chen, Z. and Liu, B. (2014). Topic modeling using topics from many domains, lifelong learning and big data. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 703–711.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 93–98.

Cohen, R., Aviram, I., Elhadad, M., and Elhadad, N. (2014). Redundancy-aware topic modeling for patient record notes. *PloS one*, 9(2):e87555.

Das, D. and Martins, A. F. (2007). A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195.

Daumé III, H. and Marcu, D. (2006). Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 305–312. Association for Computational Linguistics.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the sixth workshop on statistical machine translation*, pages 85–91. Association for Computational Linguistics.

Fan, W. and Gordon, M. D. (2014). The power of social media analytics. *Communications of the ACM*, 57(6):74–81.

Fausett, L., editor (1994). *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Fisichella, M., Stewart, A., Denecke, K., and Nejdl, W. (2010). Unsupervised public health event detection for epidemic intelligence. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1881–1884. ACM.

Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. *ACL 2017*, page 56.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM sigmod record*, pages 1–12. ACM.

He, R. and McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, abs/1602.01585.

Heinrich, G. (2008). Parameter estimation for text analysis. *University of Leipzig, Tech. Rep.*

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Hu, B., Chen, Q., and Zhu, F. (2015). LCSTS: A large scale Chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972, Lisbon, Portugal. Association for Computational Linguistics.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 168–177. ACM.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jin, W., Ho, H. H., and Srihari, R. K. (2009). A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 465–472.

Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.

Kanan, T., Zhang, X., Magdy, M., and Fox, E. (2015). Big data text summarization for events: A problem based learning course. In *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 87–90. ACM.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kruengkrai, C. and Jaruskulchai, C. (2003). Generic text summarization using local

and global properties of sentences. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 201–206. IEEE.

Law, D., Gruss, R., and Abrahams, A. S. (2017). Automated defect discovery for dishwasher appliances from online consumer reviews. *Expert Systems with Applications*, 67:84–94.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, F., Han, C., Huang, M., Zhu, X., Xia, Y.-J., Zhang, S., and Yu, H. (2010). Structure-aware review mining and summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 653–661. Association for Computational Linguistics.

Li, H., Mukherjee, A., Si, J., and Liu, B. (2015). Extracting verb expressions implying negative opinions. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2411–2417. AAAI Press.

Li, Z., Wang, B., Li, M., and Ma, W.-Y. (2005). A probabilistic model for retrospective news event detection. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 106–113. ACM.

Liang, R., Guo, W., and Yang, D. (2017). Mining product problems from online feedbacks of chinese users. *Kybernetes*, 46(3).

Lim, K. W. and Buntine, W. (2014). Twitter opinion topic model: Extracting product opinions from tweets by leveraging hashtags and sentiment lexicon. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1319–1328. ACM.

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

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating Wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Luo, X., Zhang, J., and Duan, W. (2013). Social media and firm equity value. *Information Systems Research*, 24(1):146–163.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Ly, D. K., Sugiyama, K., Lin, Z., and Kan, M.-Y. (2011). Product review summarization from a deeper perspective. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 311–314. ACM.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297. Oakland, CA, USA.

McAuley, J., Pandey, R., and Leskovec, J. (2015a). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015b). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52, New York, NY, USA. ACM.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Moghaddam, S. and Ester, M. (2011). 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. ACM.

Moghaddam, S. and Ester, M. (2012). On the design of LDA models for aspect-based opinion mining. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 803–812. ACM.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Newman, D., Noh, Y., Talley, E., Karimi, S., and Baldwin, T. (2010). Evaluating topic models for digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 215–224. ACM.

Nguyen, T.-S., Lauw, H. W., and Tsaparas, P. (2015). Review synthesis for micro-review summarization. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 169–178, New York, NY, USA. ACM.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Park, D. H., Liu, M., Zhai, C., and Wang, H. (2015a). Leveraging user reviews to improve accuracy for mobile APP retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 533–542. ACM.

Park, D. H., Zhai, C., and Guo, L. (2015b). SpecLDA: Modeling product reviews and specifications to generate augmented specifications. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM.

Paul, M. and Girju, R. (2010). A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 545–550. AAAI Press.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pitman, J. (1996). Some developments of the Blackwell-Macqueen URN scheme. *Lecture Notes-Monograph Series*, pages 245–267.

Popescu, A.-M. and Etzioni, O. (2007). Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer.

Qiao, Z., Zhang, X., Zhou, M., Wang, G. A., and Fan, W. (2017). A domain oriented lda model for mining product defects from online customer reviews. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Steinbach, M., Karypis, G., Kumar, V., et al. (2000). A comparison of document clustering techniques. In *KDD workshop on text mining*, pages 525–526. Boston.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Teh, Y. W. (2006). A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics.

Titov, I. and McDonald, R. (2008). Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Tutubalina, E. and Ivanov, V. (2014). Unsupervised approach to extracting problem phrases from user reviews of products. *COLING 2014.*, page 48.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Wang, C., Jiang, F., and Yang, H. (2017). A hybrid framework for text modeling with convolutional RNN. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2061–2069. ACM.

Wang, D., Zhu, S., Li, T., and Gong, Y. (2009). Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300. Association for Computational Linguistics.

Wang, H. and Ester, M. (2014). A sentiment-aligned topic model for product aspect rating prediction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1192–1202, Doha, Qatar. Association for Computational Linguistics.

Wang, H., Lu, Y., and Zhai, C. (2010). Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACM.

Wang, L. and Ling, W. (2016). Neural network-based abstract generation for opinions and arguments. In *Proceedings of NAACL-HLT*, pages 47–57.

Wang, S., Chen, Z., and Liu, B. (2016). Mining aspect-specific opinion using a holistic lifelong topic model. In *Proceedings of the 25th International Conference on World Wide Web*, pages 167–176. International World Wide Web Conferences Steering Committee.

Xu, W., Grishman, R., Meyers, A., and Ritter, A. (2013). A preliminary study of tweet summarization using information extraction. *NAACL 2013*, page 20.

Xue, W., Li, T., and Rishe, N. (2015). Aspect and ratings inference with aspect ratings: Supervised generative models for mining hotel reviews. In *International Conference on Web Information Systems Engineering*, pages 17–31. Springer.

Yan, Z., Xing, M., Zhang, D., and Ma, B. (2015). EXPRS: An extended PageRank method for product feature extraction from online consumer reviews. *Information & Management*, 52(7):850–858.

Yang, B. and Cardie, C. (2013). Joint inference for fine-grained opinion extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1640–1649. Association for Computational Linguistics.

Yin, D., Mitra, S., and Zhang, H. (2016). Research note–when do consumers value positive vs. negative reviews? an empirical investigation of confirmation bias in online word of mouth. *Information Systems Research*, 27(1):131–144.

Yu, Y., Duan, W., and Cao, Q. (2013). The impact of social and conventional media on firm equity value: A sentiment analysis approach. *Decision Support Systems*, 55(4):919–926.

Zaki, M. J., Meira Jr, W., and Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press.

Zhan, T.-J. and Li, C.-H. (2011). Semantic dependent word pairs generative model for fine-grained product feature mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 460–475. Springer.

Zhang, X., Niu, S., Zhang, D., Wang, G. A., and Fan, W. (2015). Predicting vehicle recalls with user-generated contents: A text mining approach. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 41–50. Springer.

Zhang, X., Qiao, Z., Tang, L., Fan, W., Fox, E., and Wang, G. (2016). Identifying product defects from user complaints: A probabilistic defect model. In *Proceedings of the 24th Americas Conference on Information Systems (AMCIS)*.

Zhang, X., Zhou, M., Fan, W., and Wang, A. G. (2014). Automatic recognition of adverse events in news: a data mining approach. In *Proceedings of Workshop on Information Technologies and Systems*.

Zhao, W. X., Jiang, J., Yan, H., and Li, X. (2010). 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. Association for Computational Linguistics.