JOINT ANALYSIS OF USER-GENERATED CONTENT AND PRODUCT INFORMATION TO ENHANCE USER EXPERIENCE IN E-COMMERCE

BY

DAE HOON PARK

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor ChengXiang Zhai, Chair
Professor Jiawei Han
Professor Kevin Chen-Chuan Chang
Assistant Professor Yi Fang, Santa Clara University

Abstract

The development of Internet has brought us a more convenient way to purchase goods through e-commerce, which has gradually pervaded our life. However, shopping experience of users in e-commerce has been far from the optimum. In order to enhance user experience in e-commerce, we propose a series of novel studies based on joint analysis of user-generated content and product information; in this dissertation, user-generated content includes user reviews and social media text data, and product information includes product descriptions and product specifications in general.

This dissertation aims at assisting e-commerce users in two directions: discovering products and making purchase decisions. To help users discover products, we first propose to leverage user reviews to improve accuracy of product search. We carefully combine product descriptions and user reviews to improve product search. Then, we also propose to recommend products via inference of implicit intent in social media text. We infer implicit intent in user status text leveraging parallel corpora we build from social media, and we recommend products whose descriptions satisfy the inferred intent.

In order to help users make purchase decisions, we first propose to generate augmented product specifications leveraging user reviews. Product specifications are often difficult to understand especially for high-technology products that contain many advanced features. We jointly model user reviews and product specifications to augment product specifications with useful information in the user reviews. We also propose to retrieve relevant opinions for new products. New or unpopular products often have no reviews, and such lack of information makes consumers hesitate to make a purchase decision. We leverage user reviews of similar products, where similarity is estimated using product specifications, to retrieve relevant opinions for new products.

The experiment results show the proposed models are effective in general. The models are also general enough to be applied to any entities with their text data. Furthermore, the models can benefit both product manufacturers and consumers, so their potential impact may be even bigger.

To my parents and my wife.

Acknowledgments

First, I must thank my advisor, Professor ChengXiang Zhai, for his guidance and inspiration throughout my entire study at UIUC. I am deeply indebted to Professor Zhai for everything I learned from him. He has always showed his passion in research, taught me how to successfully perform research, and gave me plenty of advice on both research and life. This dissertation would have been impossible without his guidance.

I would like to express my sincere gratitude to my doctoral committee members, Professor Jiawei Han, Professor Kevin Chen-Chuan Chang, and Professor Yi Fang. Their advice and suggestions on this dissertation have been invaluable and insightful.

I would like to acknowledge a debt to researchers at Caterpillar Inc., especially, Dr. Sangkyum Kim for his inestimable guidance and Dr. Christopher Ha for his continuous support. I would also like to thank researchers and interns at TCL Research America for constructive discussions and support, especially, Dr. Haohong Wang, Dr. Lifan Guo, Mengwen Liu, and Wanying Ding. I would like to thank researchers at Apple Inc. for their support, especially, Craig Mansfield, Kirk McDonald, Dr. Viktoria Rojkova, and Anshuman Mohapatra. I also would like to thank Dr. Wei Peng and Dr. Tong Sun at Xerox Research Center for their support and advice.

I would like to express my thanks to colleagues and friends for their support and help, especially, Hyun Duk Kim, Mingjie Qian, Huizhong Duan, Yue Lu, Hongning Wang, Yanen Li, Anbang Xu, Ismini Lourentzou, Maryam Karimzadehgan, Xiaolong Wang, Chase Geigle, Hoon Kim, Wooil Kim, Minje Kim, and Rui Li. I also want to thank all my colleagues in TIMAN group and DAIS group at UIUC.

Finally, I owe my deep gratitude to my parents, my sister, and my brother. Without their endless love and strong support, I would not be able to study in the United States. I would like to sincerely thank my dear wife. I would not be able to finish my doctoral study without her love and understanding.

My doctoral study was supported in part by Richard T. Cheng Endowed Fellowship. This dissertation was also supported in part by a gift fund from TCL and by the National Science Foundation under Grant Number CNS-1027965.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

With the development of Internet, the way people purchase products has gradually changed. According to United States Census Bureau, U.S. retail e-commerce sales in 2015 was estimated at about $342 billion, which is an increase of 14.6 percent from 2014.[1] More surprisingly, proportion of e-commerce sales to the total retail sales has *constantly* increased from 2.2 percent in fourth quarter of 2004 to 7.5 percent in fourth quarter of 2015 (Figure 1.1).[2] The role of e-commerce in our life becomes more and more important, and thus, we need to emphasize more on user experience in e-commerce. The shopping experience of users in e-commerce has been far from the optimum. For example, it is sometimes hard to find products a user needs, and more useful information about products in a user's perspective can be incorporated into the product information. Hence, in this dissertation, we will study how to enhance user experience in e-commerce via joint analysis of user-generated content and product information.

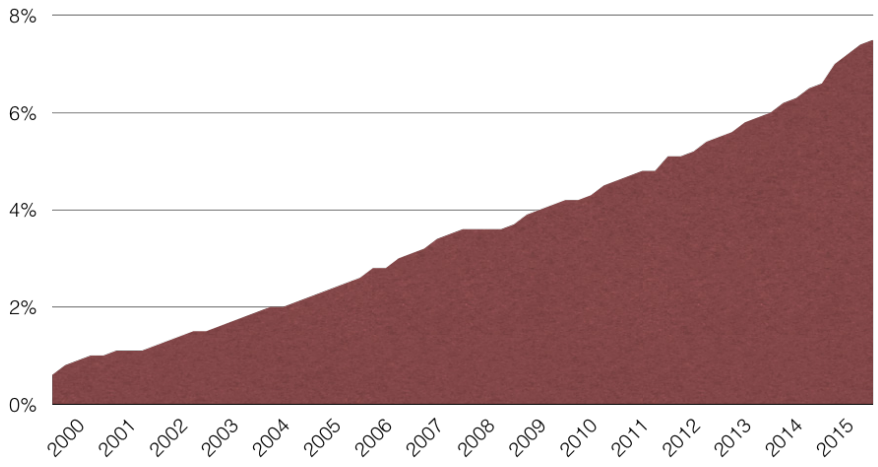


Figure 1.1: U.S. retail e-commerce sales as a percent of total retail sales.

While product information written by manufacturers helps consumers understand features of the prod-

[1] https://www.census.gov/retail/mrts/www/data/pdf/ec_current.pdf
[2] Estimates are adjusted for seasonal variation.

ucts, user-generated content such as user reviews can offer indirect experiences to consumers. Thankfully, the growth of e-commerce resulted in an abundance of user reviews. Online retailers have encouraged consumers to leave reviews on products in order to help other consumers make purchase decisions. Indeed, according to a survey, 60 percent of consumers preferred to purchase products from a site that has user reviews, and 48 percent of consumers considered user reviews more influential than advertising (24 percent) or recommendation from sales assistants (22 percent).[3] At the same time, people leave numerous status messages on social media with the development of Internet. Everyday, 500 million tweets are left on Twitter.[4] People express their needs in various ways on social media, so social media text can be a good resource to understand user needs for e-commerce. In order to provide better user experience in e-commerce, we thus need to take a closer look at user-generated content such as user reviews and social media text.

Meanwhile, online retailers usually provide product information such as product descriptions and product specifications. Such product information helps consumers understand the products well so that they can easily make purchase decisions. Product descriptions describe product aspects in an unstructured or weakly structured form while product specifications describe technical characteristics of products in a structured form. While product descriptions are essential in e-commerce to describe products with easy words, online retailers often provide product specifications with technical terms. Product specifications are provided at more than 50 percent of surveyed online retailers,[5] and they consist of a set of (feature, value) pairs. Information in product specifications is useful especially for high-technology products with several electronic components because they enable consumers to understand the details of products in an organized way. The abundance of such data including product descriptions and specifications enables us analyze them to mine useful knowledge.

Both user-generated content and product information can be important resources to enhance user experience in e-commerce. However, while text mining on either user-generated content or product information has received much attention, there are a limited number of studies that jointly analyze user-generated content and product information (especially product specifications) despite their big potential impact.

1.2 Challenges

Although user-generated content and product information are valuable resources to enhance user experience in e-commerce, the usefulness of independent analysis on each of them is limited. For example, opinion mining and sentiment analysis [76] on user reviews of a product can be useful to customers only if there

[3] https://www.reevoo.com/news/half-of-consumers-find-social-content-useful-when-shopping-online/
[4] http://www.internetlivestats.com/twitter-statistics/
[5] https://www.marketingsherpa.com/article/chart/product-page-display

exists any reviews for the product; such techniques cannot provide any analysis for products that do not have any reviews. Therefore, we need techniques that can jointly analyze user-generated content and product information to provide even more useful knowledge. For example, we can jointly analyze user reviews and product specifications of existing products to retrieve relevant opinion sentences for new products. However, there are challenges in the joint analysis.

1.2.1 Vocabulary Gap between User-Generated Content and Product Information

There exists a vocabulary gap between user-generated content and product information. User-generated content such as user reviews and social media text is written by ordinary users, and it often contains lots of informal expressions. On the other hand, product information such as product descriptions and product specifications is written by manufacturers, and it is usually written with formal expressions. Moreover, while general users focus more on the product aspects they consider, manufacturers often write every detail of the products. The resulting vocabulary gap lets the resources have different characteristics such as Inverse Document Frequency (IDF) of words, which is considered very important in analyzing text data. Therefore, we should carefully handle the vocabulary gap when we jointly analyze the different kinds of resources.

1.2.2 Joint Analysis with Structured Product Information

Previously, researchers mostly studied independently on user reviews. However, user reviews can be even more useful together with structured product information such as product specifications. For example, product specifications, which consist of (feature, value) pairs, are often difficult for novice users to understand. Instead of listing (feature, value) pairs of a product, users may want additional guidance that helps them understand the product features. For example, from the specifications, users may want to find out which features are more important in a user's perspective. Users may also want to know what are special about each product. In addition, it may be useful if users can view opinions relevant to certain features or feature values when they view the specifications.

Such data mining tasks require a joint analysis of user reviews and product specifications. The associations between product features, which are often standardized, and consumer opinions may be very interesting to product manufacturers as well as users. For example, manufacturers can find out which features of their product are liked or disliked by customers so that they can improve their products based on user opinions. Therefore, mining opinions with consideration of product specifications would provide us very useful analysis. However, while many researchers studied independent analysis on user reviews, only few researchers studied

joint analysis of user reviews and product specifications, and the joint analysis requires special techniques to control two different text data: unstructured text (user reviews) and structured text (product specifications).

1.2.3 Noise and Mixed Opinions in User-Generated Content

User-generated content is often very noisy. User reviews and social media text are written by ordinary users who often ignore grammar and make typos. Kaufmann [44] describes that tweets contain so much noise that it is difficult to extract useful information from them. On the other hand, user reviews about a product often diverge in their opinions. Sometimes, users write consistent opinions about the product, but users often have contradictory opinions about it. Thus, it is more desirable to find opinions that are central among all opinions of a product. In addition to such mixed opinions, user-generated content often contains spam texts that may harm credibility of the analysis. We thus need to remove the noise and handle the mixed opinions well to provide a more credible analysis.

1.3 Joint Analysis of User-Generated Content and Product Information to Enhance User Experience in E-commerce

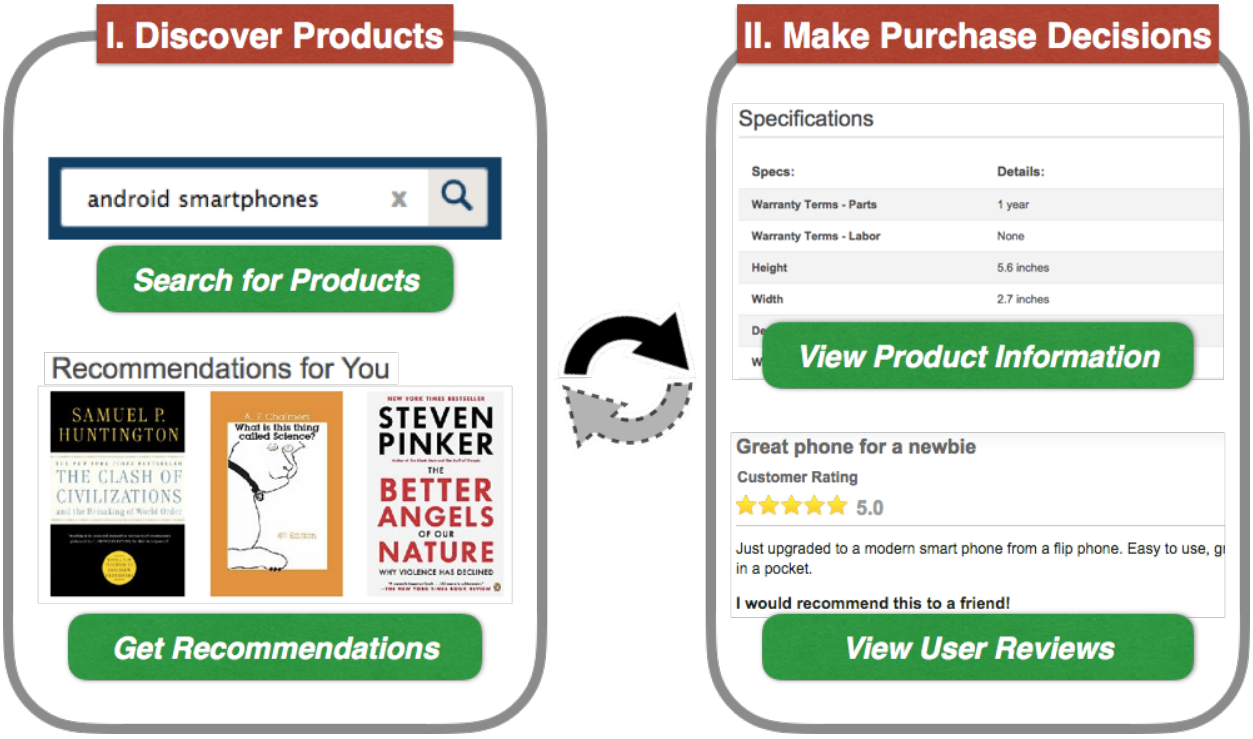


Figure 1.2: A typical shopping experience in e-commerce. This dissertation studies each component in the left and the right parts.

A typical shopping experience in e-commerce is depicted in Figure 1.2. A user first discovers a product via product search or recommendation, which may be personalized or not. Once the user finds a relevant product, the user reads the provided product information in order to make a purchase decision. If the user is still interested, the user may further read user reviews to find out what other customers have said about the product. After digesting enough information, the user decides whether to purchase the product or not. If the user does not like the product, then the user may go back to the previous step, where the user can discover other products.

In other words, we assume that there are two main roles of online retailers: assisting users in discovering products and in making purchase decisions. To help users discover products, we can let the users type keywords to search for products they are interested in, or more proactively, we can recommend products that the users may be interested in. In this dissertation, we propose two directions to assist users in discovering products via joint analysis of user-generated content and product descriptions: (1) product search leveraging user reviews and (2) product recommendation via inference of implicit intention in social media text. Once users find a product they are interested in, e-commerce sites are supposed to provide detailed information about the product in order to help them make purchase decisions. In this dissertation, we propose additional two directions to generate useful information about products via joint analysis of user reviews and product specifications: (3) generating augmented specifications and (4) opinion retrieval for new products.

In specific, we propose four directions in total to achieve our goal:

1. We propose to leverage user reviews to improve accuracy for product search, especially mobile app search. Mobile apps are now important in our everyday life, where average American consumers spend more than two hours per day inside the apps. Meanwhile, the number of mobile apps in app stores has explosively increased. The myriad apps have made it extremely hard for consumers to discover needed apps without search or recommendation. While there are a few commercial mobile app search engines available, the new task of mobile app retrieval has not yet been rigorously studied. We first study the effectiveness of the existing general information retrieval models. Then, we propose a novel approach that jointly models app descriptions and user reviews. Our key idea is to leverage user reviews to find out important features of apps and bridge vocabulary gap between app developers and users. We also create a test data set for evaluating the novel problem. Experiment results indicate that the proposed approach effectively models app descriptions and user reviews and outperforms state-of-the-art retrieval models.

2. We propose to recommend products, *i.e.*, mobile apps, for social media users. People often implicitly or explicitly express their needs in social media in the form of "user status text", and such text can

be very useful for service providers and product manufacturers to proactively provide relevant services or products that satisfy people's immediate needs. We study how to infer a user's intent based on the user's "status text" and recommend mobile apps that may satisfy the user's need. We address this problem by framing it as a new entity retrieval task where the query is a user's status text and the entities to be retrieved are the mobile apps. We first propose a novel approach that generates a new representation for each query. Our key idea is to leverage user-generated content, *i.e.*, social media text, to build parallel corpora that contain implicit intention text and the corresponding explicit intention text. Specifically, we model various user intentions in social media text using topic models, and we predict the user intention in the query that contains implicit intention. Then, we retrieve relevant mobile apps with the predicted user intention. We evaluate the task using a new data set we create. Experiment results indicate that the proposed model is indeed useful to understand user intentions and outperforms the state-of-the-art retrieval models.

3. We propose to generate augmented specifications by jointly modeling user reviews and product specifications. Product specifications provide organized details of a product, and such information can help users make purchase decisions. While product specifications are often available at online retailers, they are not straightforward for novice consumers to understand, especially advanced features of high-technology products. We jointly model user reviews and product specifications to generate augmented product specifications, which can help consumers understand products and their features. In specific, we propose a novel Specification Latent Dirichlet Allocation (SpecLDA) that can enable us to effectively model user reviews and product specifications at the same time. The augmented specifications inform customers what other customers have said about the features in the reviews of the same product and also different products. SpecLDA can also infer importance of each feature and infer which words are special for each product so that customers can quickly understand a product they are interested in. Experiment results show that SpecLDA can effectively augment product specifications with useful knowledge from user reviews.

4. We propose to retrieve relevant opinion sentences for new products. With the rapid development of E-commerce, abundant user reviews have been written by consumers who bought the products. These reviews are very useful for consumers to optimize their purchase decisions. While user reviews are abundant for popular products, the majority of products still lack user reviews since they are new or unpopular. For such products with no reviews, we propose to exploit user reviews of other products that are similar to them. Our key idea is to leverage product specifications to assess similarity between the query product and other products and extract relevant opinion sentences from reviews of the similar

products, where a consumer may find useful discussions. Then, we provide ranked opinion sentences for the query product that has no user-generated reviews. We create a new data set and propose an evaluation method that can automatically evaluate the retrieved review sentences without manual efforts. Experiment results show that our novel probabilistic methods can effectively retrieve useful opinion sentences for products that have no reviews.

The rest of this dissertation is organized as follows. The common related work is discussed in Chapter 2. In Part I, we discuss how to assist users in discovering products. We first propose to leverage user reviews for product search in Chapter 3. Then, we present the study of product recommendation via inference of implicit intention in social media text in Chapter 4. In Part II, we discuss how to assist users in making purchase decisions via mining useful knowledge. We first present the study of generating augmented specifications by jointly modeling user reviews and product specifications in Chapter 5. Then, we propose to retrieve relevant opinion sentences for new products through joint analysis of user reviews and product specifications in Chapter 6. We discuss the conclusion and promising future directions in Chapter 7.

Chapter 2

Related Work

The development of Internet has brought e-commerce, a new way to purchase a product, to humanity. The growth of e-commerce attracted many researchers' attention. For example, to understand why people shop online, Swaminathan *et al.* [93] studied what factors influence online purchasing behavior. Their results imply that reliability of vendors, competitive prices, useful information, and ease of canceling orders affect the frequency of online purchases. They also found that consumers who are motivated by convenience are more likely to shop online. To further understand e-commerce consumers, Rohm and Swaminathan [88] developed a typology of online shoppers based on shopping motives such as online convenience, physical store orientation, information use in planning and shopping, and variety seeking in the online shopping context. Researchers also have studied how shopping experience in e-commerce can be enhanced. For example, Häubl and Trifts [31] developed interactive tools such as a recommendation agent and a comparison matrix that help e-commerce consumers make much better decisions while making substantially less effort.

In this dissertation, we jointly analyze user-generated content and product information to enhance e-commerce shopping experience. Here, we introduce some related work that either jointly or independently analyzes user-generated content and product information for e-commerce. In general, user reviews and social media text are used as user-generated content, and product descriptions and specifications are used as product information.

Text mining on user-generated content for products has been widely studied with the development of e-commerce, in order to help consumers make purchase decisions. For example, user reviews are one of the most important online resources to obtain other consumers' opinions on a product. It has been shown that user reviews have a significant impact on sales of products such as movies and hotels [24, 108]. However, the great amount of reviews make consumers hard to digest them, resulting in demand for automatic opinion mining techniques. Opinion mining such as opinion retrieval and summarization on user reviews thus attracted a lot of attentions for a decade. There are several surveys which summarize the existing opinion mining work [76, 60, 45]. Opinion mining has different characteristics from general text mining. In opinion mining, subjective text (opinions) is focused while objective text is often excluded. Such subjective text is often

analyzed to identify its opinion polarity (*e.g.*, positive or negative), which readers care about. Also, targets of opinions are generally identified since opinions are expressed towards targets such as products or aspects of products. Thus, aspect-based opinion mining [37, 85, 101] has been studied as a main stream in the field.

Likewise, user reviews have been employed to retrieve and summarize opinions for products. There are also a few studies that employed user reviews to predict ratings [75, 29] or sales [20] of a product. For example, Pang and Lee [75] proposed to infer an author's implied numerical rating of a review. However, user review analysis basically cannot be performed if there does not exist any user reviews for a product. We thus study how to retrieve relevant opinion sentences for products that do not have any user reviews. As far as we know, there is no existing work that studied to retrieve opinions for such products; previous studies mine opinions for products that already have reviews.

Most of the studies leveraging user-generated content for e-commerce focused on opinion mining and summarization. Despite their abundance, user reviews have been leveraged for product search by only a few researchers. Ganesan and Zhai [28] proposed opinion-based entity ranking leveraging user reviews. They ranked entities such as products based on a user's preferences expressed in reviews. For example, for a query "fantastic battery life", their ranking method retrieved products with good battery life according to the reviews. However, they exploited only user reviews to represent products, and the product information was not exploited. Duan *et al.* [22] leveraged product specifications as well as user reviews to improve performance on product search. While they jointly analyze product specifications and user reviews for product search, we jointly analyze product descriptions and user reviews, which may be more general since products in some categories such as mobile apps do not have product specifications.

Meanwhile, product information such as product specifications have been exploited for product ranking and search. For example, ranking techniques for structured product database were proposed in [15, 92]. However, although product specifications have been available in many e-commerce sites, only a few studies employed them for joint analysis with user reviews. For example, Zhou and Chaovalit [117] performed sentiment classification on user reviews leveraging domain ontology database from IMDb[1]. Other related studies employed product specifications to build an aspect hierarchy [112] and summarize product features [103, 82]. However, none of them studied feature values as well as features in specifications. In addition, to the best of our knowledge, no research work has studied the problem of augmenting product specifications. We jointly analyze user reviews and product specificaitons to augment product specificaitons.

[1] `http://www.imdb.com`

Part I

Assisting Users in Discovering Products

Chapter 3

Leveraging User Reviews to Improve Accuracy for Product Retrieval[1]

3.1 Introduction

Myriads of products make it difficult for consumers to find "right products" for them. There can be mainly two ways to help consumers discover products. We can let users submit a query, and the product search engine can retrieve products relevant to the query. More proactively, we can recommend products that the consumers are likely to be interested in. While product recommendation can suggest products to consumers more proactively than product search, it is more difficult for product recommendation to identify the immediate and explicit needs of consumers, which means that product search and product recommendation have their own advantages and disadvantages. In this chapter, we study how to leverage user reviews to improve accuracy for product retrieval. In the next chapter, we will study how to identify a user's explicit intention in the user's social media text.

Product search is a kind of entity search, and it is different from Web search in that there are often user reviews for products while there are very few user reviews for Web documents. In this work, we focus on mobile apps in the product domain since mobile apps occupy a large share of our everyday life, and there exist many user reviews for them. According to a recent analysis by Flurry, average American consumers spend about three hours (177 minutes) per day on mobile devices,[2] which is more than the average time spent on TVs (168 minutes). An analysis in 2013 shows that 80% of the time spent on mobile devices is inside apps and 20% is spent on the mobile web.[3] The time spent on the mobile web remained flat in 2014 while the time spent inside apps increased. While consumers spend much of their time inside apps, they constantly download new mobile apps.[4] This means that the role of app search and recommendation system remains important.

Meanwhile, the number of mobile apps in app store is explosively increasing so that the search function in app store becomes essential. As of July 2014, there are about 1.3 million apps in Google Play app store

[1] Part of this work has been published in [78].
[2] http://www.flurry.com/blog/flurry-insights/mobile-television-we-interrupt-broadcast-again
[3] http://www.flurry.com/bid/95723/Flurry-Five-Year-Report-It-s-an-App-World-The-Web-Just-Lives-in-It
[4] http://www.flurry.com/blog/flurry-insights/app-install-addiction-shows-no-signs-stopping

and 1.2 million apps in Apple App Store.[5] The myriad apps made consumers extremely hard to find apps without search or recommendation functions. For example, Google Play does not list all of the apps. Instead, it only lists recommended or popular apps because finding an app through the long list does not make sense any more. Moreover, in an app developer's view, new or unpopular apps are barely discovered by consumers if they are not recommended by the app stores. Therefore, app search engine is definitely essential for both consumers and developers.

Thus, it is our goal to find apps based on a query given by a user. Specifically, given a user query that describes an aspect of an app, the desired search result would show a ranked list of apps where higher ranked apps are more likely to have the described aspect. For example, for a query "book a flight", we expect the search result to include apps such as "Expedia Hotels & Flights" and "Orbitz – Flights, Hotels, Cars" in high ranks since these apps meet the user's need quite well. However, if an app description is not written well, *e.g.*, too short or hardly useful, the retrieval system would not rank the app high even though the app is actually relevant to a query. In addition, app descriptions are written by app developers while search queries are made by users, and this results in vocabulary gap between them. Therefore, to improve accuracy for mobile app retrieval, we propose to leverage user reviews, which provide more information about an app in the user's vocabulary.

Although it is an interesting new retrieval task, app retrieval has not yet been rigorously studied in the literature. Indeed, no test collection has ever been created yet for quantitatively evaluating this task. To address this problem, we conduct the first systematic study of effectiveness of both existing retrieval models and new retrieval models for this task, and we create the very first test collection for evaluating this new retrieval problem.

This work makes the following contributions:

1. We introduce and study a novel research problem of mobile app retrieval leveraging user reviews. To the best of our knowledge, this is the first effort in this area, and no other research work studied the same problem as ours.

2. We propose a novel probabilistic topic model that jointly models user reviews and unstructured product information (product description) in order to obtain representation of apps. The model captures topics jointly from reviews and descriptions so that the topics reflect vocabulary of both users (user reviews) and developers (app descriptions). The model is unsupervised and general, so it can be applied to other domains where there are unstructured text about an entity and an associated set of unstructured text.

[5] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

3. We create a new data set for evaluating the task and conduct experiments to show that the proposed method outperforms baseline approaches for mobile app retrieval. The mobile app data set is crawled from a major app store, Google Play. We let domain experts generate queries based on android forums. Then, we collect query relevance data via crowdsourcing service. The test collection is available at `http://timan.cs.uiuc.edu/downloads.html`. As far as we know, no previous research for mobile app retrieval has performed quantitative evaluations.

3.2 Related Work

Recommendation systems are highly related to information retrieval systems in that they rank objects to fulfill needs of users, and the recommendation systems are surveyed well in [38]. App recommendation systems have been studied by a few researchers [118, 111]. For example, Lin *et al.* [58] addressed cold start problem in app recommendation system by leveraging social network data. However, retrieval systems are different from recommendation systems mainly because a user explicitly expresses his or her needs in retrieval systems while the recommendation systems suggest items based on user profile without asking for the user's needs. Recommendation systems may be more convenient for users since a user does not have to input his or her needs, but they are likely to be less accurate than information retrieval systems since they barely know the user's immediate needs. In addition, recommendation systems encounter a cold start problem when a user does not have his or her profile yet or when the system does not have enough transaction data yet. On the other hand, information retrieval systems do not require such data, so there is no cold start problem for them.

There have been extensive studies for XML retrieval, which is related to our work since app data consist of elements such as app descriptions and user reviews. Some of the XML retrieval studies also support simple keyword queries [62] while some other XML retrieval studies support only structured queries [100]. However, our goal is to augment app descriptions with user reviews to obtain a better representation for an app, while XML retrieval focuses more on document structure in general. In addition, while retrieval unit in our task is clearly defined as a mobile app, in XML retrieval, every element is a retrievable unit [42], which makes XML retrieval different from our task.

Entity ranking and entity search are closely related to our problem [19]. While entity ranking usually focuses on exploiting rich structures [98, 81], Ganesan and Zhai [28] studied opinion-based entity ranking leveraging only review text. In their work, a query is structured with preferences on different aspects of an entity. The known-type entities are then ranked based on aspect-based sentiment from user reviews, while

we rank unknown-type apps based solely on query relevance. Product search is highly related to our work. Duan *et al.* [22] leveraged product specifications and user reviews to improve performance on keyword search in product database. However, while products usually have such structured specifications that characterize products, mobile apps usually have no structured specifications since there is no standardized features of apps, which makes the problem more challenging. Meanwhile, there have been some efforts to build commercial app search engines such as [17]. However, mobile app retrieval problem has not yet been studied rigorously with systematic experiments.

3.3 Problem Definition

In order to find an app, a user constructs a text query q, where we assume q represents the search intent of the user, and q is input to an app retrieval system. Then, the app retrieval system searches for apps that satisfy the user intent and shows the user a list of apps that are ranked according to their relevance to q, which conforms to the probability ranking principle [87]. Formally, we are given M apps $\boldsymbol{A} = \{a_1, ..., a_M\}$. For each app a_i, there is an unstructured app description d_i and user reviews that are concatenated to a single review document, r_i. Our goal is to retrieve a list of apps for each q based on their descriptions and/or reviews and rank them in order of the probability of relevance. Figure 3.1 illustrates our problem.

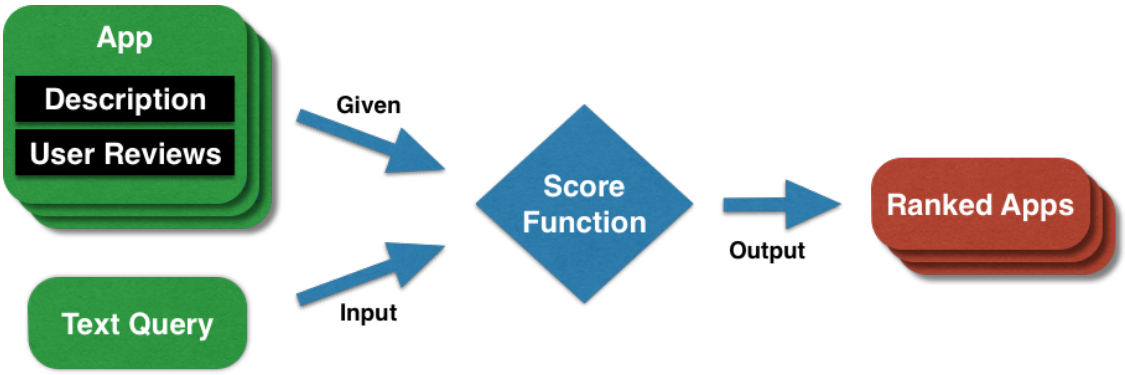


Figure 3.1: Mobile app retrieval with app descriptions and user reviews.

To the best of our knowledge, retrieval of entities exploiting opinionated content as well as entity description is a new problem that has not been well addressed yet in previous work. Kavita and Zhai [28] ranked entities leveraging user reviews but no associated descriptions. In [27] and [22], the researchers exploited user reviews and associated structured data to rank entities while we do not use structured data but unstructured description data about entities.

User reviews are good extra sources to find apps especially when an app description is too short or is

poorly described. However, leveraging both app descriptions and user reviews is challenging because those two types of unstructured text are written by different authors in different views. Consequently, different topics are stressed and different vocabulary sets are used in the two types of data, which make them difficult to combine. In addition, user reviews often contain content that does not address the entity's features; indeed, a huge portion of reviews is about installation problems or general sentiment on the whole app. Therefore, careful unification of the two different types of data is desired.

The main research questions we would like to answer in our study are:

1. *How can we create a test collection for evaluating this new retrieval task?* When query log data is not available, obtaining realistic queries is challenging and important for research.

2. *How effective are the existing general retrieval models for this task?* Since there is no previous work studied on this task, we do not know how well existing models will work for app retrieval.

3. *Can reviews help?* User reviews can be easily obtained at app stores, but it is unknown whether app retrieval systems can be improved by leveraging them.

4. *Can topic models be used to more effectively combine descriptions and reviews?* Given two different types of text data, how can we effectively generate unified representation of descriptions and reviews?

3.4 Test Set Creation

Although mobile apps have pervaded our everyday life, there does not exist a test data set for mobile app retrieval task. Thus, in this section, we discuss how we can create the test set. In particular, we discuss how we can collect mobile apps from commercial app stores, where we can obtain realistic queries, and how we can collect query relevance data.

3.4.1 Collecting Apps from App Stores

App descriptions and user reviews are available at multiple app stores such as Google Play, Apple App Store, and Amazon Appstore. Among them, we chose Google Play because it is one of the largest app stores with abundant reviews. In Google Play app store, there are 41 categories of apps, where each app is assigned with only one category. From each category, we crawled about one thousand popular apps on average, which are somehow ranked by Google, resulting in about 43 thousand apps in total. For each app, we crawled up to the first 50 user reviews, which are ranked by their helpfulness votes. To compare our methods with the search engine in Google Play, we also crawled top 20 retrieved apps and their reviews from Google Play for

each search query. After all, we crawled 43,041 app descriptions (one description per app) and 1,385,607 user reviews in total (32.2 reviews per app on average). We pre-processed text in the descriptions and reviews in the following way. We first tokenized text into word tokens and lemmatized them using Stanford CoreNLP [65] version 1.3.5. We lowered word tokens and removed punctuation. Then, stopwords and word tokens that appear in less than five descriptions and five reviews were removed. Also, word tokens that appear in more than 30 percent of descriptions or reviews were removed since they do not carry important meanings. We finally have 18,559 unique word tokens (V), and the statistics of the resulting text data is shown in Table 3.1.

Table 3.1: Statistics of text data in app descriptions ($\boldsymbol{D}$) and user reviews ($\boldsymbol{R}$).

	$\boldsymbol{D}$	$\boldsymbol{R}$
Average number of tokens	94.1	176.4
Total number of tokens	4,051,366	7,592,779

3.4.2 Where Can We Obtain Realistic Queries?

In order to quantitatively evaluate how well the suggested methods perform, we need a query set and a query relevance data set. Unfortunately, there does not yet exist such test collection. We thus create our own test collection. However, collecting realistic queries that embed the needs of users for app search is not easy. Researchers who do not work for companies that provide an app search engine generally do not have access to the query log data, hence it is hard for them to know which apps users want to find and which apps are difficult to find. To collect such real queries, we propose to leverage an app forum.

There is an Android forum[6] where users frequently ask various kinds of questions about Android including Android apps. We employ Google search engine to find threads containing an exact phrase "looking for an app" in the forum in order to find posts about app search. This can be done by giving the following query to the Google search engine: *"looking for an app" site:forums.androidcentral.com*. Then, for each search result, we systematically determined to collect the post or not. We retained a post only if the user looks for an app and the thread includes one or more answers that recommend relevant apps because there are users who look for non-existing apps. The first sixty such posts were saved, and one of them with title "walkie talkie app" is shown below.

> I'm looking for an app that if I push a button and choose either of my kids android phones. I want it to just play on their phone without having to click on a voice file or do something interactive. Kind of like an intercom. Does anything like this exist?

[6] http://forums.androidcentral.com/

Next, we asked domain experts to write a short search query (usually a couple of keywords) for each post, by pretending they were the authors who wrote those posts and want to search for the app at app stores. Examples of such generated queries are: "locate cell tower", "podcast streamer", "nightstand clock", "auto text while driving", and "music player for church". Please note that the collected queries may not precisely reflect representative queries in actual app stores, and collecting such queries is left as our future work. Meanwhile, one may be concerned that the queries are biased towards difficult ones since we obtain them from forum posts, where users post questions when they do not know the answers. The relevance data in the next section show that the queries are not "very" difficult.

3.4.3 Collecting Query Relevance Data

To judge whether a retrieved app is relevant to a query, we need human-labeled relevance data. However, labeling all the retrieved apps by humans is too expensive. We thus created a pool, which consists of top retrieved apps from different retrieval systems, and we employed a crowdsourcing service, CrowdFlower[7], to label them at affordable prices. Specifically, for each query, we pooled together the top 20 retrieved apps from each of the suggested methods with multiple parameter combinations. Then, for each of the (query, retrieved app) pairs, we made a question providing the short query, the entire post, and the link of the retrieved app, and we asked three annotators to label it. Each annotator was asked to read the query and entire question post, follow the link to the app store, read the app description, and then judge if the app satisfies the search intent on three relevance levels (no satisfaction at all (0), partial satisfaction (1), and perfect satisfaction (2)).

Collecting a high-quality gold standard data set through crowdsourcing is often difficult since there are lots of abusers. To control quality of the relevance judgments, we manually judged relevance of 120 (query, app) pairs and used them as quiz questions. Each annotator was allowed to start labeling the data set only if the annotator passes our quiz session with at least 75% of accuracy, where the quiz session consists of eight randomly selected quiz questions. We also inserted a random quiz question for every four (query, app) pairs in a random order, without telling which one is a quiz question. If the annotator's accuracy goes below 75% at any point of time, we removed all the answers from the annotator and asked other annotators to judge them. We paid five cents for each judgment, and each annotator was limited to judge up to 250 questions. When all the needed judgments were made, we verified the given links to the app store, and if the links are no longer valid, we removed the (query, app) pairs since the annotators may have made random answers when they encountered the broken links. The three resulting judgments for each (query, app) pair were

[7] http://www.crowdflower.com/

averaged to be used as a relevance score. From the 60 queries, we discarded four queries that Google Play search engine could not retrieve relevant apps in top 10 apps.

Table 3.2: Statistics of relevance data for 56 queries. Some statistics include standard deviations followed by "±".

Avg. # of words in each query	4.04 ± 1.43
# of distinct (query, app) pairs judged	4,534
# of all judgments	13,602
Avg. # of (query, app) pairs for each query	81.0 ± 17.5
# of all annotators	272
Avg. # of judgments for each annotator	50.0 ± 48.1
Fleiss' kappa	0.39
Perfect agreement rate	0.71

The statistics of relevance data for the resultant 56 queries are shown in Table 3.2. To measure inter-annotator agreement, we employed Fleiss' kappa. The kappa value is 0.39, which can be interpreted as between "Fair agreement" and "Moderate agreement" according to [50]. Perfect agreement rate, which is the proportion of (query, app) pairs where all three annotators agree on judgment, is 0.71. To see how difficult each query is, we counted the number of judgments where at least two annotators judged as perfect relevant. For each query, there are 13.6 such relevant apps on average with standard deviation being 7.87, which means that the queries are not very difficult considering the size of the data set. This may be due to the fact that the forum post was uploaded a while ago so that the non-existing apps could have been released before we crawl the data set.

3.5 Methods

In order to retrieve apps that best match a query q, we first try existing standard retrieval models based only on app descriptions, which is a typical information retrieval problem. Then, we add user reviews to the data set to see if they are useful for the task. To combine app descriptions with user reviews, we propose a topic model-based method as well as traditional methods.

3.5.1 Standard Text Retrieval

Despite the importance of app retrieval problem, it has not been answered how well standard text retrieval methods perform. We, therefore, employ existing state-of-the-art methods here.

Okapi BM25

The Okapi BM25 method has been one of the state-of-the-art methods for ad-hoc retrieval. As presented in [25], BM25 scores a document d with respect to q as follows:

$$score(q,d) = \sum_{w \in q \cap d} [\frac{(k_3+1)c(w,q)}{k_3+c(w,q)} \times \frac{(k_1+1)c'(w,d)}{k_1+c'(w,d)} \times \log \frac{N+1}{df(w)+0.5}] \tag{3.1}$$

where $c(w,q)$ is w's count in q, $df(w)$ is a document frequency of w in a corpus, N is the number of all documents in a corpus, and k_1, k_3, and b are parameters. A normalized count of w in d, $c'(w,d)$, is defined as

$$c'(w,d) = \frac{c(w,d)}{1-b+b\frac{N_d}{avl(d)}} \tag{3.2}$$

where $c(w,d)$ is w's count in d. N_d is the length of d, $avl(d)$ is the average length of d in a corpus. We use this model as one of the most popular text retrieval methods.

Query Likelihood Language Model (QL)

Query Likelihood retrieval model was introduced by Ponte and Croft in [84] using multiple Bernoulli to model documents. Instead of multiple Bernoulli, most researchers have focused on using multinomial to model documents since it was shown to perform better than multiple Bernoulli model [91]. Therefore, we use Query Likelihood method with multinomial model (unigram language model) in this work. Query Likelihood scores a document d with respect to q as follows:

$$score(q,d) = \prod_{w \in q} p(w|d) \tag{3.3}$$

where $p(w|d)$ is a probability of w being in d. In order to avoid over-fitting and keep $p(w|d)$ from being zero, $p(w|d)$ is smoothed by Dirichlet smoothing technique and defined as

$$p(w|d) = \frac{N_d}{N_d+\mu} p_{ml}(w|d) + \frac{\mu}{N_d+\mu} p(w|\boldsymbol{D}) \tag{3.4}$$

where $\boldsymbol{D}$ is a set of all documents, and $p_{ml}(w|d)$ and $p(w|\boldsymbol{D})$ are estimated by maximum likelihood estimator (MLE), yielding $p_{ml}(w|d)=\frac{c(w,d)}{\sum_{w'} c(w',d)}$ and $p(w|\boldsymbol{D})=\frac{c(w,\boldsymbol{D})}{\sum_{w'} c(w',\boldsymbol{D})}$. Smoothing parameter μ enables the system to dynamically smooth $p_{ml}(w|d)$ based on the length of d, N_d. Consequently, Query Likelihood Language Model with Dirichlet smoothing is regarded as one of the state-of-the-art retrieval models, and we call it QL in this work. Please refer to [114] for more information on smoothing language models.

Topic Model-based Approach

Traditional retrieval models such as BM25 and QL do not consider association among words, which makes the system unable to retrieve documents that do not contain a query word. If there exists a vocabulary gap between queries and documents, the retrieval system is not supposed to work well. To solve the problem, we focus on enriching document representation with topic models. Please note that techniques such as query expansion can be combined with our suggested methods.

A topic model is a probabilistic model that can find latent themes and their distributions in a document from a text collection, where a theme (topic) is a cluster of words whose occurrence in documents overlap frequently. Thus, even if a document d does not contain a certain word w, $p(w|d)$ can be high enough if d contains many words that are in the same topic as w. For example, even if a word "bistro" is not contained in a description for a restaurant finder app, the app can be retrieved if the description contains a word "restaurant" since the two words are likely to be in the same topic(s). The two most popular topic models are Probabilistic Latent Semantic Analysis (PLSA) [32] and Latent Dirichlet Allocation (LDA) [8]. PLSA has two main problems: (1) the number of parameters grows as the data set size grows, and (2) it does not generate a new document, which has not been seen in training data. Those problems are solved in LDA by utilizing Dirichlet allocation, and thus, we employ LDA in this work.

For app retrieval problem, we suggest to exploit LDA-based document model (LBDM) [105], which has been shown to effectively model documents. The LBDM-based retrieval system was shown in [109] to generally outperform a retrieval system with a more sophisticated topic model, Pachinko Allocation Model (PAM) [55], which captures correlations among topics. Thus, we employ LBDM as one of the baselines in this study. We still use the same scoring formula as in (3.3), where the document language model $p(w|d)$ is replaced with LBDM. As presented in [105], $p(w|d)$ of LBDM involves a linear interpolation of MLE-estimated language model and LDA document model, which is defined as

$$p(w|d) = \lambda\big[\frac{N_d}{N_d+\mu}p_{ml}(w|d) + \frac{\mu}{N_d+\mu}p(w|\boldsymbol{D})\big] + (1-\lambda)p_{lda}(w|d) \tag{3.5}$$

where the LDA document model, $p_{lda}(w|d)$, is described in [105] in detail. As in equation (3.4), MLE-estimated document model, $p_{ml}(w|d)$, is smoothed with MLE-estimated corpus model, $p(w|\boldsymbol{D})$.

3.5.2 Retrieval with Descriptions and Reviews

App descriptions are not written perfectly by app developers. For example, some descriptions are too short, and some others may contain too much useless information for search. Luckily, abundant user reviews are

available, which may be a good source to complement such descriptions. Another important reason to leverage user reviews is that both search queries and user reviews are written from a user's perspective while descriptions are written from a developer's perspective. Due to the nature of apps, app descriptions are usually written mainly about their features. However, app developers may not exactly know what terms users would like to use to describe the features. For example, an app description may contain a phrase "find nearby restaurants" to describe its feature. If a user searches for "food near me", which does not have any common terms with the description, the app will not be retrieved by simple keyword matching even though the two phrases are about the same feature. In such case, user reviews may play an important role to bridge vocabulary gap between app developers and users. If there is a user review containing a phrase such as "good app for locating food near me" and the retrieval system indexes the review as well, the app would be retrieved even when the description does not have such terms.

To leverage user reviews, we need to somehow combine representations of a description d and a concatenated user review r. Combining representations of two different data sets by simply adding words in them together may not be a good idea if the data sets have different characteristics. In this section, we describe how to combine them using our novel method as well as traditional methods.

BM25F

BM25F has been known as the state-of-the-art for structured information retrieval. Regarding descriptions and reviews as different fields of a document, we can apply BM25F to our problem. Similar to [83], we replace $c'(w,d)$ in equation (3.1) with $c''(w,a)$, which is defined as

$$c''(w,a) = \frac{boost_d \cdot c(w,d)}{1 - b_d + b_d \frac{|d|}{avl(d)}} + \frac{boost_r \cdot c(w,r)}{1 - b_r + b_r \frac{|r|}{avl(r)}} \tag{3.6}$$

where $boost_d$ and $boost_r$ are weights for d and r, respectively, and b_d and b_r play the same role as b does for BM25. $|r|$ is a length of review r, and $avl(r)$ is the average length of r in a review corpus.

Combined Query Likelihood

To combine two different types of text data, it may be better to assign some portion of an app representation to description data and some other portion of it to user review data. Thus, the unigram language model for a description and a review, $p(w|d)$ and $p(w|r)$, respectively, can be combined as in [73] to build a unified language model for an app, $p(w|a)$, which is defined as

$$p(w|a) = (1-\eta)p(w|d) + \eta p(w|r) \tag{3.7}$$

where η is a parameter to determine the proportion of review language model for $p(w|a)$. To score apps with respect to q, we follow the score function of QL. $p(w|d)$ and $p(w|r)$ are estimated by MLE and smoothed as in QL, and the resulting score function for q and a is defined as

$$\begin{aligned} score(q,a) &= \prod_{w \in q} p(w|a) \\ &= \prod_{w \in q} \big[(1-\eta)p(w|d) + \eta p(w|r)\big] \\ &= \prod_{w \in q} \Big[(1-\eta)\Big(\frac{N_d}{N_d+\mu_d} p_{ml}(w|d) + \frac{\mu_d}{N_d+\mu_d} p(w|\boldsymbol{D})\Big) \\ &\quad + \eta\Big(\frac{N_r}{N_r+\mu_r} p_{ml}(w|r) + \frac{\mu_r}{N_r+\mu_r} p(w|\boldsymbol{R})\Big)\Big] \end{aligned} \tag{3.8}$$

where $p(w|\boldsymbol{R})$ is a review corpus language model, N_r is the number of words in r, and μ_d and μ_r are Dirichlet smoothing parameters for d and r, respectively.

AppLDA: a topic model for app descriptions and user reviews

In our task, the role of topic model is similar to that of user reviews in that they both provide augmentation of vocabulary. In addition to bridging vocabulary gap, we design a topic model that can also remove noise in reviews. The key idea is to simultaneously model app descriptions and user reviews by sharing topics between the two different types of text and discarding parts of reviews if they don't share topics with app descriptions. Intuitively, when a user writes a review, the user would decide if he or she writes about a topic in app description or some other topics such as installation problems. Assuming that those other topics (review-only topics) do not help us retrieve relevant apps, we remove review texts that are about review-only topics in order to have a better estimation of app representations. We thus form review-only topics as well as shared topics to filter out review texts that are not useful for retrieval. Figure 3.2 illustrates the different kinds of topics.

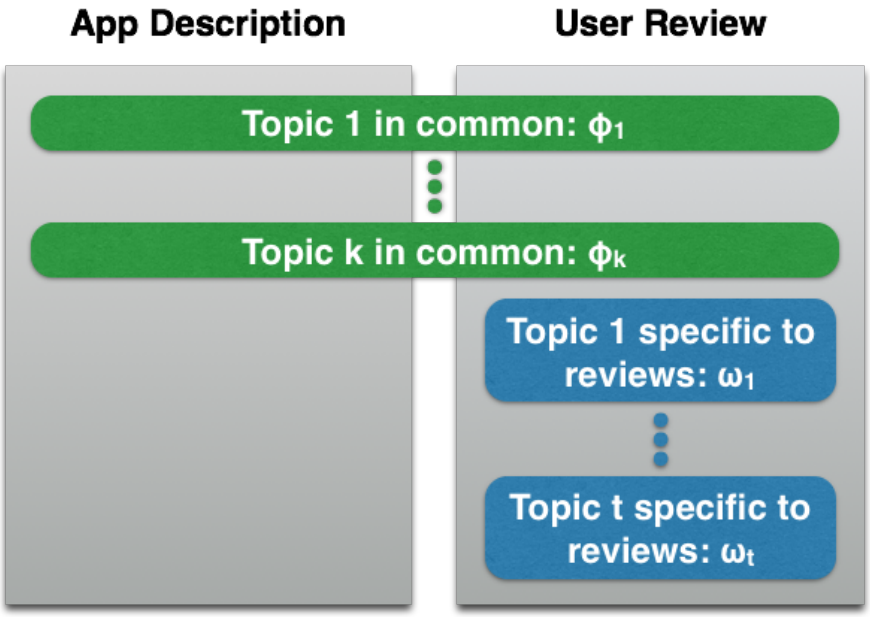


Figure 3.2: Shared topics and review-only topics in app descriptions and user reviews.

Algorithm 1 Generative Process of AppLDA

for each shared topic z **do**
  draw $\boldsymbol{\phi}_z \sim Dirichlet(\boldsymbol{\beta})$
end for
for each review topic y **do**
  draw $\boldsymbol{\omega}_y \sim Dirichlet(\boldsymbol{\gamma})$
end for
for each app a with a description d and a review r **do**
  draw $\boldsymbol{\theta}_d \sim Dirichlet(\boldsymbol{\alpha}^d)$
  for each $i \in \{1, \ldots, N_d\}$ **do**
    draw $z_{d,i} \sim Multi(\boldsymbol{\theta}_d)$
    draw $w_{d,i} \sim Multi(\boldsymbol{\phi}_{z_{d,i}})$
  end for
  draw $\boldsymbol{\psi}_r \sim Beta(\boldsymbol{\delta})$
  draw $\boldsymbol{\theta}_r \sim Dirichlet(K \cdot \boldsymbol{\alpha}^p \cdot prior(\boldsymbol{\alpha}^d, \boldsymbol{z}_d) + \boldsymbol{\alpha}^r)$
  draw $\boldsymbol{\pi}_r \sim Dirichlet(\boldsymbol{\tau})$
  for each $i \in \{1, \ldots, N_r\}$ **do**
    draw $x_{r,i} \sim Bernoulli(\boldsymbol{\psi}_r)$
    if $x_{r,i} = 0$ **then**
      draw $z_{r,i} \sim Multi(\boldsymbol{\theta}_r)$
      draw $w_{r,i} \sim Multi(\boldsymbol{\phi}_{z_{r,i}})$
    else
      draw $y_{r,i} \sim Multi(\boldsymbol{\pi}_r)$
      draw $w_{r,i} \sim Multi(\boldsymbol{\omega}_{y_{r,i}})$
    end if
  end for
end for

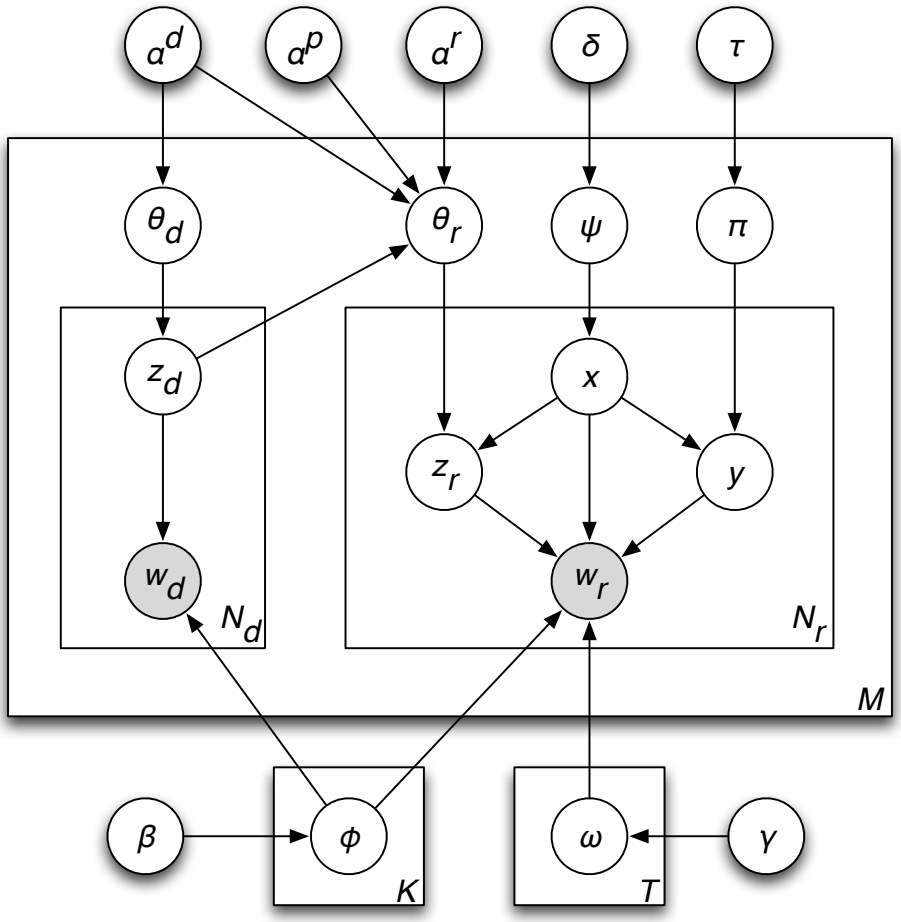


Figure 3.3: Graphical representation of AppLDA.

The graphical representation of AppLDA is depicted in Figure 3.3, and its generative process is described in Algorithm 1. The generation of app description by an app developer can be regarded as a typical topic modeling process that is explained for regular LDA in earlier this section. After an app description is generated, each word $w_{r,i}$ of review r with length N_r for an app a is written by a user. The user first chooses whether to write about topics that are shared with descriptions or some other topics that are far from the shared topics using switch $x_{r,i}$ according to a Bernoulli distribution $\boldsymbol{\psi}_a$, which is drawn from a Beta distribution with a symmetric tuple $\boldsymbol{\delta}$. If shared topics are chosen ($x_{r,i} = 0$), the user further specifies a shared topic $z_{r,i}$ from the topic distribution in r, $\boldsymbol{\theta}_r$, which is drawn from a Dirichlet distribution with an asymmetric vector $K \cdot \boldsymbol{\alpha}^p \cdot prior(\boldsymbol{\alpha}^d, \boldsymbol{z}_d) + \boldsymbol{\alpha}^r$. Here, K is the number of all shared topics, and $\boldsymbol{\alpha}^p$ is a symmetric vector. $prior(\boldsymbol{\alpha}^d, \boldsymbol{z}_d)$ is a distribution generated from topics in d, which is estimated by $\frac{N_{z,d}+\alpha^d}{N_d+K\alpha^d}$, where N with subscription and/or superscription means the number of words satisfying subscription/superscription conditions. For example, $N_{z,d}$ means the number of words assigned with z in d, and N_d is the number of words in d. Then, the user writes a word $w_{r,i}$ about the chosen shared topic according to a multinomial word distribution $\boldsymbol{\phi}_{z_{r,i}}$, which is drawn from a Dirichlet distribution with a symmetric vector $\boldsymbol{\beta}$. On the other hand, if the user chooses to write about topics that are far from shared topics ($x_{r,i} = 1$), the user further chooses a review topic $y_{r,i}$ according to a multinomial topic distribution $\boldsymbol{\pi}_r$, which is drawn from a Dirichlet distribution with a symmetric vector $\boldsymbol{\tau}$. Then, $w_{r,i}$ is chosen according to a word distribution $\boldsymbol{\omega}_{y_{r,i}}$, which is drawn from a Dirichlet distribution with a symmetric vector $\boldsymbol{\gamma}$. This process is repeated for all words in all app descriptions and user reviews for I iterations. Please note that all values in a symmetric vector are the same; *e.g.*, $\boldsymbol{\alpha} = \{\alpha, ..., \alpha\}$.

In order to guide the model to learn hidden topics in reviews, we use prior knowledge from topic distribution in app descriptions by $prior(\boldsymbol{\alpha}^d, \boldsymbol{z}_d)$. Intuitively, when a user writes a review about shared topics, the distribution of shared topics in reviews is likely to be at least somewhat similar to that in app descriptions. For example, if an app description is about finding nearby restaurants, the reviews are more likely to contain topics regarding restaurants than other topics such as finance or game topics. The prior knowledge in app descriptions is thus passed to reviews in the form of asymmetric prior distribution, $prior(\boldsymbol{\alpha}^d, \boldsymbol{z}_d)$, and this distribution is referred to draw topics in reviews. Here, the strength of the prior knowledge is controlled by the symmetric vector $K \cdot \boldsymbol{\alpha}^p$, and the prior knowledge is smoothed with the symmetric vector $\boldsymbol{\alpha}^r$. In other words, we can view this process as follows. A user is given a set of topics in an app description, and the user writes a review about the app referring to the topics in the description. Such prior knowledge can be employed via imposing asymmetric priors on the topic distributions of reviews. More information on applying asymmetric priors in a topic model can be found in [99].

The collapsed Gibbs sampling formulas to learn latent variables z_d, z_r, x, and y for an app a are as follows. Learning a topic of the ith word in d, $z_{d,i}$, is defined as

$$\begin{aligned} p(z_{d,i}|\boldsymbol{W}^d, \boldsymbol{Z}^d_{\setminus d,i}, \boldsymbol{\alpha}^d, \boldsymbol{\beta}) &\propto p(w_{d,i}|z_{d,i}, \boldsymbol{W}^d_{\setminus d,i}, \boldsymbol{Z}^d_{\setminus d,i}, \boldsymbol{\beta}) p(z_{d,i}|\boldsymbol{Z}^d_{\setminus d,i}, \boldsymbol{\alpha}^d) \\ &\propto \frac{N^{\setminus d,i}_{w_{d,i}|z_{d,i}} + \beta}{N^{\setminus d,i}_{z_{d,i}} + V\beta} \times \frac{N^{\setminus d,i}_{z_{d,i}|d} + \alpha^d}{N_d - 1 + K\alpha^d} \end{aligned} \tag{3.9}$$

where $\boldsymbol{W}^d$ is a set of all words in the description corpus, $\boldsymbol{Z}^d$ is all shared-topic assignments for those words in all descriptions, V is the size of vocabulary $\boldsymbol{V}$, and K is the number of all shared topics. Again, N with subscription and/or superscription means the number of words satisfying subscription/superscription conditions, and "$\setminus d, i$" means excluding d's ith data. To learn a shared topic ($x_{r,i} = 0$) for the ith word in r, $z_{r,i}$, we define the Gibbs sampling formula as

$$\begin{aligned} &p(x_{r,i} = 0, z_{r,i}|\boldsymbol{W}^r, \boldsymbol{Z}^r_{\setminus r,i}, \boldsymbol{Z}^d, \boldsymbol{X}_{\setminus r,i}, \boldsymbol{\alpha}^d, \boldsymbol{\alpha}^r, \boldsymbol{\alpha}^p, \boldsymbol{\delta}, \boldsymbol{\beta}) \\ &\quad\propto p(x_{r,i} = 0|\boldsymbol{X}_{\setminus r,i}, \boldsymbol{\delta}) \times p(w_{r,i}|z_{r,i}, \boldsymbol{W}^r_{\setminus r,i}, \boldsymbol{Z}^r_{\setminus r,i}, \boldsymbol{\beta}) \times p(z_{r,i}|\boldsymbol{Z}^r_{\setminus r,i}, \boldsymbol{Z}^d, \boldsymbol{\alpha}^d, \boldsymbol{\alpha}^r, \boldsymbol{\alpha}^p) \\ &\quad\propto \frac{N^{\setminus r,i}_{x=0|r} + \delta}{N_r - 1 + 2\delta} \times \frac{N^{\setminus r,i}_{w_{r,i}|z_{r,i}} + \beta}{N^{\setminus r,i}_{z_{r,i}} + V\beta} \times \frac{N^{\setminus r,i}_{z_{r,i}|r} + K\alpha^p \frac{N_{z_{r,i}|d} + \alpha^d}{N_d + K\alpha^d} + \alpha^r}{\left(\sum_z N^{\setminus r,i}_{z|r}\right) + K(\alpha^p + \alpha^r)} \end{aligned} \tag{3.10}$$

where $\boldsymbol{W}^r$ is all words in the review corpus, and $\boldsymbol{Z}^r$ is all shared-topic assignments for those words in all reviews. On the other hand, to learn a review-only topic ($x_{r,i} = 1$) for the ith word in r, $y_{r,i}$, we define the Gibbs sampling formula as

$$\begin{aligned} &p(x_{r,i} = 1, y_{r,i}|\boldsymbol{W}^r, \boldsymbol{Y}_{\setminus r,i}, \boldsymbol{X}_{\setminus r,i}, \boldsymbol{\tau}, \boldsymbol{\delta}, \boldsymbol{\gamma}) \\ &\quad\propto p(x_{r,i} = 1|\boldsymbol{X}_{\setminus r,i}, \boldsymbol{\delta}) \times p(w_{r,i}|y_{r,i}, \boldsymbol{W}^r_{\setminus r,i}, \boldsymbol{Y}_{\setminus r,i}, \boldsymbol{\gamma}) \times p(y_{r,i}|\boldsymbol{Y}_{\setminus r,i}, \boldsymbol{\tau}) \\ &\quad\propto \frac{N^{\setminus r,i}_{x=1|r} + \delta}{N_r - 1 + 2\delta} \times \frac{N^{\setminus r,i}_{w_{r,i}|y_{r,i}} + \gamma}{N^{\setminus r,i}_{y_{r,i}} + V\gamma} \times \frac{N^{\setminus r,i}_{y_{r,i}|r} + \tau}{\left(\sum_y N^{\setminus r,i}_{y|r}\right) + T\tau} \end{aligned} \tag{3.11}$$

where $\boldsymbol{Y}$ is a set of review-only topic assignments for all words in all reviews, and T is the number of all review-only topics.

Retrieval with AppLDA In order to retrieve apps relevant to a query q, we need document representations for apps, so we create a unigram language model for each a, $p_{lda}(w|a)$, which is defined as

$$
\begin{aligned}
p_{lda}(w|a) &= \sum_{z=1}^{K} p(w|z, \boldsymbol{W}^d, \hat{\boldsymbol{Z}}^d, \boldsymbol{\beta}) p(z|a, \hat{\boldsymbol{Z}}^d, \hat{\boldsymbol{Z}}^r, \boldsymbol{\alpha}^d, \boldsymbol{\alpha}^r, \boldsymbol{\alpha}^p) \\
&\propto \sum_{z=1}^{K} \frac{\hat{N}_{w|z} + \beta}{\hat{N}_z + V\beta} \times \frac{\hat{N}_{z|d} + \alpha^d + \hat{N}_{z|r} + K\alpha^p \frac{\hat{N}_{z|d} + \alpha^d}{N_d + K\alpha^d} + \alpha^r}{N_d + K\alpha^d + \left(\sum_z \hat{N}_{z|r}\right) + K(\alpha^p + \alpha^r)}
\end{aligned}
\tag{3.12}
$$

where $\hat{\boldsymbol{Z}}^d$ and $\hat{\boldsymbol{Z}}^r$ are topics for descriptions and reviews estimated from AppLDA, respectively, and $\hat{N}$ with subscription is the estimated number of words satisfying the subscription condition. The formula can be interpreted as the unification of LDA-estimated language models for descriptions and reviews, where the words that are not assigned with the shared topics are removed. In other words, the description and the cleaned review form a single unified document for each app, and the unified language model is used for retrieval. The AppLDA-estimated language model is combined with the MLE-estimated language models to define the score function for q and a as follows:

$$
\begin{aligned}
score(q, a) &= \prod_{w \in q} p(w|a) \\
&= \prod_{w \in q} \Big((1-\lambda) p_{lda}(w|a) + \lambda \big[\frac{N_d + N_{x=0|r}}{N_d + N_{x=0|r} + \mu} p_{ml}(w|a) + \frac{\mu}{N_d + N_{x=0|r} + \mu} p(w|\boldsymbol{A}) \big] \Big)
\end{aligned}
\tag{3.13}
$$

where $N_{x=0|r}$ is the number of words assigned with shared topics in reviews, and $p_{ml}(w|a)$ is MLE-estimated language model for a's description and cleaned review, which is defined as

$$
p_{ml}(w|a) = p(w|a, \boldsymbol{W}^d, \boldsymbol{W}^r, \hat{\boldsymbol{X}}) \propto \frac{N_{w|d} + N_{x=0,w|r}}{N_d + N_{x=0|r}} \tag{3.14}
$$

and $p(w|\boldsymbol{A})$ is estimated by MLE for descriptions and cleaned reviews of all apps $\boldsymbol{A}$, and it is defined as

$$
p(w|\boldsymbol{A}) = p(w|\boldsymbol{A}, \boldsymbol{W}^d, \boldsymbol{W}^r, \hat{\boldsymbol{X}}) \propto \frac{N_{w|\boldsymbol{D}} + N_{x=0,w|\boldsymbol{R}}}{N_{\boldsymbol{D}} + N_{x=0|\boldsymbol{R}}} \tag{3.15}
$$

and μ is a Dirichlet smoothing parameter for MLE-estimated language models, and λ is a weight for MLE-estimated language models against the topic model-estimated language model. In order to estimate reliable values for LDA estimated language models, it is recommended to use multiple Markov chains in [105]. Similar to the results in [105], we found that three Markov chains with 100 Gibbs sampling iterations each show reasonably reliable performance, so we follow this setting.

3.6 Experiments

In this section, we first describe how we set parameters for the experiments. Then, we qualitatively analyze the search results from the suggested methods, and we quantitatively evaluate the results using our test collection.

3.6.1 Parameter Setting

The following parameter values are used in the experiments unless otherwise specified. The parameters are tuned from our data set based on average of four NDCG measures specified in section 3.6.3. For BM25, we use the standard value k_3=1,000, and we set the parameters k_1=4.0 and b=0.4, which showed the best performance. For BM25F, we tune the parameters at our best, and we consider the following values: k_3=1,000, k_1=3.5, b_d=0.4, b_r=0.3, $boost_d$=0.6, and $boost_r$=0.4. For Query Likelihood Language Model (QL), μ is tuned to be 1,000. For Combined Query Likelihood (CombQL), the same μ is used, and we set μ_r=300 and η=0.4, which showed the best performance. For LBDM, K is tuned to be 300, and we set topic model parameters $\alpha=\frac{50}{K}$ and β=0.01, which is the common setting in the literature. Its retrieval parameters λ and μ are tuned to be 0.5 and 1,000, respectively. To see how well regular QL and LBDM perform with both data sets $\boldsymbol{D}$ and $\boldsymbol{R}$, we simply add words in reviews to the corresponding descriptions and used the merged documents as an input to QL and LBDM; we call these methods as QL($\boldsymbol{D}, \boldsymbol{R}$) and LBDM($\boldsymbol{D}, \boldsymbol{R}$). μ for QL($\boldsymbol{D}, \boldsymbol{R}$) is tuned to be 800. For LBDM($\boldsymbol{D}, \boldsymbol{R}$), K is tuned to be 300, and the same α and β values are used as for regular LBDM, and retriever parameters λ and μ are tuned to be 0.5 and 800, respectively. For our proposed model AppLDA, we set K=300 and T=30, which showed the best performance. We use the standard values for other topic model parameters: $\alpha^d=\alpha^r=\frac{50}{K}$, $\tau=\frac{50}{T}$, and $\beta=\gamma$=0.01. If one believes that the reviews have a specific amount of shared-topic proportion, then δ can be used as asymmetric prior. However, we let the model fully figure out the proportions, so we set δ=0.5 for symmetric vector $\boldsymbol{\delta}$, which is a small value. A larger value of $alpha^p$ lets the distribution of shared-topics in reviews be more similar to that in app descriptions; α^p is tuned to be 0.05. For retrieval parameters of AppLDA, we set λ=0.5 and μ=800, which showed the best performance.

3.6.2 Qualitative Analysis

Table 3.3 and 3.4 show the biggest shared topics and review-only topics (measured by $\hat{N}_z$), respectively, estimated by AppLDA. At Google Play, 18 of 41 categories are game-related categories, so they are reflected in the topics; for example, the biggest shared topic is about "casino game", and the fourth review-only topic

Table 3.3: Top shared topics by AppLDA.

slot	check	photo	news
bonus	deposit	picture	article
win	mobile	pic	story
machine	account	gallery	break
spin	bank	image	read
casino	balance	effect	local
coin	banking	editing	latest
payout	credit	editor	content
slots	transfer	filter	fox
jackpot	transaction	edit	live

Table 3.4: Top review-only topics by AppLDA.

log	interface	ad	addictive
account	uus (ui)	pop	pass
login	design	annoying	addicting
sign	function	remove	enjoy
error	miss	advert	addict
password	lack	advertisement	challenge
website	user	rid	kill
connect	functionality	full	entertaining
server	improvement	seconds	interesting
access	clean	click	challenging

is about "sentiment towards games". The other biggest shared topics are about "mobile banking", "photo editor", and "news article", and each of them represents a feature of an app well. Review-only topics seem reasonable since the words in them are likely to appear more often in reviews than in descriptions. We also compare top review-only topics from AppLDA with top topics from regular LDA when we use only user reviews, which is shown in Table 3.5. Comparing the biggest topics of them, which are both about "account", it is shown that the topic in LDA is corrupted with "bank"-related words such as "bank" and "deposit" while the topic in AppLDA is about general accounts of apps. An "account problem" topic is more likely to appear

Table 3.5: Top topics in the reviews by LDA.

log	upgrade	purchase	note
account	battle	refund	support
check	spend	upgrade	pro
mobile	character	reinstall	developer
login	gold	bug	sync
password	rpg	force	program
sign	gameplay	year	tool
deposit	attack	month	dev
service	level	lose	tablet
bank	fight	block	draw

in review-only topics while a "mobile banking" topic is more likely to appear in shared topics. AppLDA is able to separate such topics well by forming two different types of topics. Interestingly, AppLDA's shared-topics consist of words that do not carry sentiment while review-only topics often contains words carrying sentiment such as "miss", "lack", "annoying", and "addicting". This means that AppLDA separated the two different types of topics reasonably well; since regular LDA does not explicitly separate them, top topics in reviews contain few sentiment words, but several words of them are likely to appear often in app descriptions.

To show the difference of retrieved apps from different models, we retrieve apps for a query "locate cell tower" using the suggested methods. The top retrieved apps are shown in Table 3.7. According to the question post, the query looks for an app that locates a cell tower the phone is currently attached to. By comparing methods that do not leverage user reviews (QL and LBDM) with methods that do leverage user reviews (CombQL and AppLDA), we can see the effect of adding more review text. The relevant apps such as "Map My Cell Tower" and "zBoost Signal Finder" do not contain the query word "locate", which makes them hard to find. However, since the reviews of those apps contain phrases such as "Won't even locate my cell tower" and "Handy app to locate towers", CombQL and AppLDA could rank them high. While the reviews help bridge vocabulary gap by adding absent words from review text, topic model-based method also bridges vocabulary gap by connecting associated words. LBDM gave high scores to relevant apps such as "zBoost Signal Finder", "Map My Cell Tower", and "Signal Finder", which do not contain the word "locate" in their reviews, even though it does not leverage user reviews. Since the descriptions of those apps contain words such as "gps" and "map" that are in the same topics as "locate" is, they could be ranked high.

3.6.3 Quantitative Analysis

Table 3.6: NDCG evaluation results for mobile app retrieval. The first three methods exploit only app descriptions, $\boldsymbol{D}$, while the next five methods leverage user reviews, $\boldsymbol{R}$, as well as app descriptions, $\boldsymbol{D}$.

	N@3	N@5	N@10	N@20
BM25	0.578	0.550	0.527	0.537
QL	0.541	0.517	0.511	0.515
LBDM	0.584	0.563	0.543	0.565
BM25F	0.597	0.596	0.583	0.597
QL ($\boldsymbol{D}$,$\boldsymbol{R}$)	0.613	0.618	0.585	0.586
CombQL	0.637	0.624	0.602	0.593
LBDM($\boldsymbol{D}$,$\boldsymbol{R}$)	0.610	0.625	0.613	0.626
AppLDA	**0.651**†§	**0.656**†‡	**0.627**†	**0.634**†‡
Google Play	0.589	0.575	0.568	0.566

Since we average relevance judgments of three annotators, the aggregated relevance is defined as a real number in [0,2]. Hence, evaluation metrics such as Mean Average Precision (MAP), which requires binary

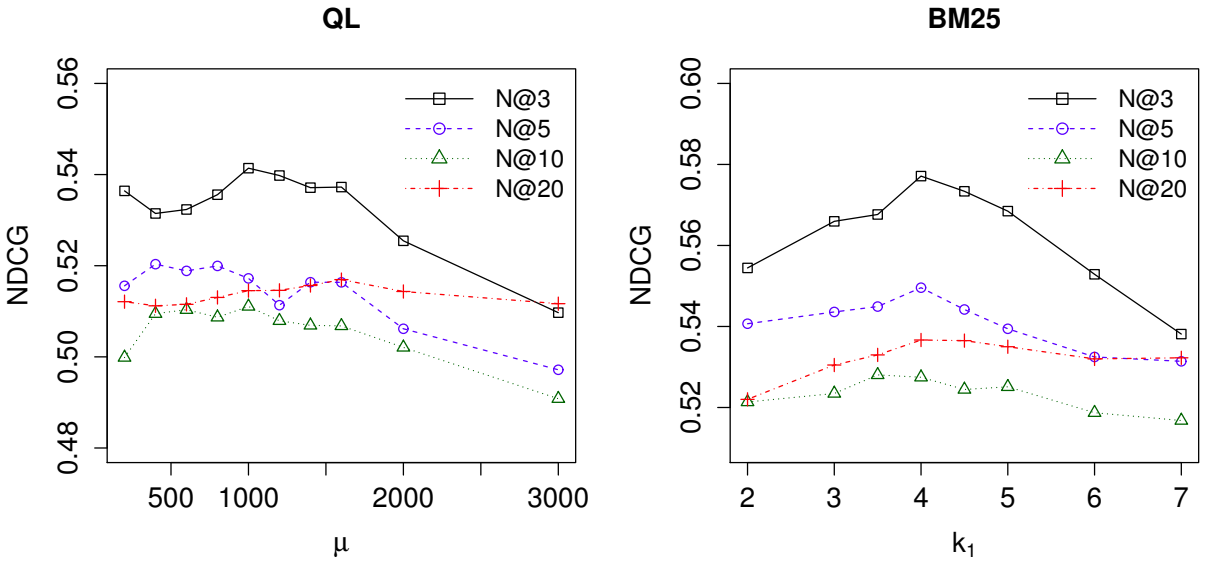


Figure 3.4: NDCG measures for different μ values of QL (left) and for different k_1 values of BM25.

Table 3.7: Top retrieved apps for a query "locate cell tower". Strikethrough indicates irrelevant apps.

QL	LBDM	CombQL	AppLDA
~~GPS Tracker Pro~~	zBoost Signal Finder	Cell Map	Cell Map
Tower Collector	3G 4G WiFi Map & Speedtest	zBoost Signal Finder	Map My Cell Tower
3G 4G WiFi Map & Speedtest	~~GPS Tracker Pro~~	Signal Finder	zBoost Signal Finder
~~Locate -Find Friends & Family~~	Tower Collector	Network Signal Info Pro	Signal Finder
~~Find My Phone~~	Map My Cell Tower	Tower Collector	Network Signal Info Pro
inViu OpenCellID	~~Locate -Find Friends & Family~~	Map My Cell Tower	Tower Collector
~~Signal Booster Reloaded~~	inViu OpenCellID	3G 4G WiFi Map & Speedtest	Cell Info Display
~~Signal Booster for Android~~	Signal Finder	~~Family Tracker: Locate Phones~~	3G 4G WiFi Map & Speedtest
~~Family Tracker: Locate Phones~~	~~Find My Phone~~	inViu OpenCellID	~~Sprint Family Locator~~
~~Digital Leash Locate~~	~~Family Tracker: Locate Phones~~	~~GPS Tracker Pro~~	~~GPS Tracker Pro~~

relevance, cannot be employed. We instead employ Normalized Discounted Cumulative Gain (NDCG) [40] as the evaluation metric because NDCG is able to measure ranking performance on multiple-level relevance data. Specifically, we measure NDCG at 3, 5, 10, and 20 top retrieved apps to reflect diverse users' information needs. Unlike traditional web search, NDCG@3 might be quite important; it is common for app stores to show only one or a couple of retrieved apps on a smartphone screen. On the other hand, obtaining judgments for top 20 retrieved apps of retrieval systems with all combinations of parameter values is too expensive and not realistic in practice. We judged top 20 retrieved apps of the suggested retrieval systems with 22 parameter value combinations. The relevance data may be thus incomplete. However, it is shown in [110, 11] that ignoring unjudged documents is effective in such situations. Therefore, we alternatively employ induced NDCG at k judged apps, which shares the same philosophy as induced MAP in [110]. Induced NDCG ignores unjudged apps from the ranked list and is calculated in the same way as regular NDCG. We simply call it NDCG in this work.

We compare the performance of the suggested methods as well as Google Play's app search engine. Table 3.6 shows the evaluation results. †, ‡, and § are used to mark if the improvement for AppLDA is statistically (paired t-test with $p \leq 0.05$) significant in each measure over LBDM, CombQL, and LBDM($\boldsymbol{D}$,$\boldsymbol{R}$), respectively. As expected, LBDM outperforms BM25 and QL in all measures when only app descriptions are available because LBDM is able to recognize semantically related words. When only app descriptions are used, BM25 outperforms QL in all measures, and their performance on different parameter values is shown in Figure 3.4. However, when both descriptions and reviews are used, QL($\boldsymbol{D}$,$\boldsymbol{R}$) and CombQL outperform BM25F in all measures except NDCG@20, which means QL and CombQL are more suitable to combine different data types for app retrieval than BM25F. It is clear to see that the models that leverage user reviews perform better than the models that use only descriptions. For example, BM25F outperforms BM25, and CombQL and QL($\boldsymbol{D}$,$\boldsymbol{R}$) outperform QL with a relatively big performance difference. In addition, AppLDA and LBDM($\boldsymbol{D}$,$\boldsymbol{R}$) outperform LBDM, which means that topic model's capability of bridging vocabulary is even amplified when user reviews are added. QL($\boldsymbol{D}$,$\boldsymbol{R}$) exploits user reviews by concatenating descriptions and user reviews, and it is outperformed by CombQL that combines description model and user review model with linear interpolation. This means that the review data set and description data set have their own characteristics, so they need to be combined without losing them. AppLDA outperforms CombQL and LBDM($\boldsymbol{D}$,$\boldsymbol{R}$), and the improvement is statistically significant in two measures and one measure, respectively. While CombQL does not score high in NDCG@10 and NDCG@20, LBDM($\boldsymbol{D}$,$\boldsymbol{R}$) does not score high in NDCG@3. AppLDA seems to complement such drawbacks of CombQL and LBDM($\boldsymbol{D}$, $\boldsymbol{R}$) by effectively modeling app descriptions and user reviews.

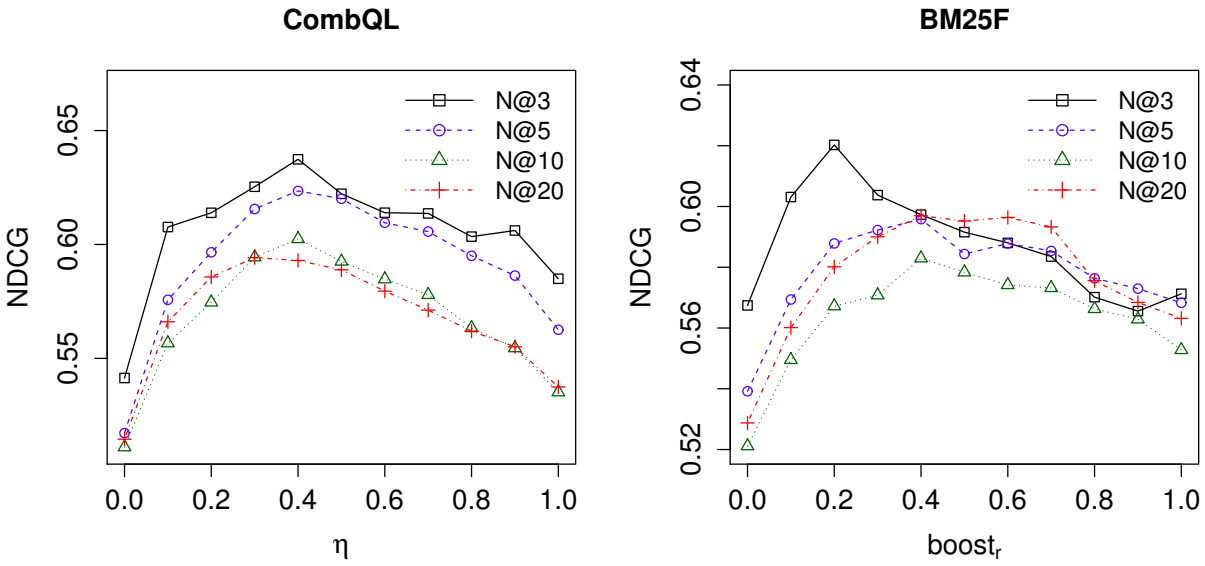


Figure 3.5: NDCG measures for different review weights (η) of CombQL (left) and for different review weights ($boost_r$) of BM25F when $boost_d = 1.0 - boost_r$ (right).

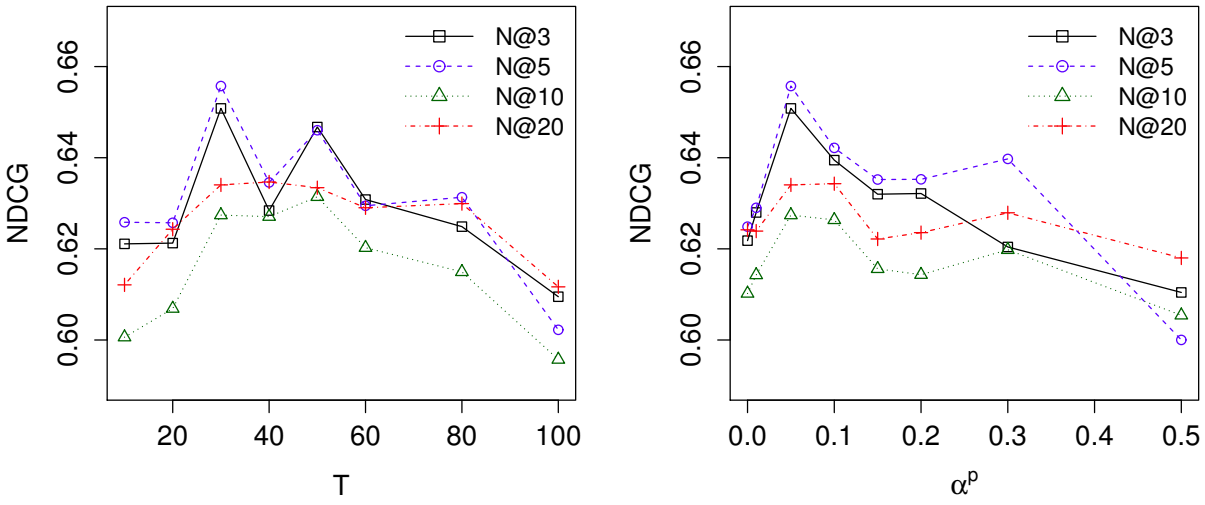


Figure 3.6: NDCG measures for AppLDA when different numbers of review-only topics (T) are used (left) and different prior weights (α^p) are used (right).

In order to understand the effects of leveraging reviews, we further investigate performance when different proportions of review representations are used. NDCG measures for different η values of CombQL and different $boosts_r$ values of BM25F are shown in Figure 3.5. η is a weight of review language model in CombQL, and $boost_r$ is a weight of normalized count of word in reviews. Here, we set $boost_d = 1.0 - boost_r$ for BM25F. When $\eta = 1.0$ or $boost_r = 1.0$, the models exploit only reviews while they exploit only descriptions when $\eta = 0.0$ or $boost_r = 0.0$. Surprisingly, for both models, using only reviews gives a better performance than using only descriptions; which means that review data set may be a better resource for app search than description data set. Since both reviews and queries are written by users, there may be less vocabulary gap between them, resulting in reviews being more useful for app retrieval than descriptions. Combining app descriptions and user reviews yields even better performance, which peaks when $\eta = 0.4$ and $boost_r$ is around 0.4. This means that descriptions and reviews supplement each other well.

Figure 3.6 shows AppLDA's performance when different numbers of review-only topics T are used and different values of α^p priors are used. It seems that about 30 to 50 review-only topics exist in the review data set, and setting too few or too many review topics harm the performance. The number of review-only topics is much smaller than that of shared topics (300). This is because there are various available features for each category of apps while the topics users mention other than app features converge to a small set of topics such as "installation problems" and "user interface". Meanwhile, α^p controls the amount of topic distribution priors for user reviews, and it is obtained from topic distribution of app descriptions. The priors are used to give clues when identifying topics in reviews under the assumption that the distribution of shared topics in reviews is likely to be similar to that of app descriptions. Indeed, it is shown in Figure 3.6 that adding priors is helpful while adding too much priors is rather harmful. The performance peaks at $\alpha^p = 0.05$ when $K = 300$, which means that giving about fifteen guiding words to a noise-removed review is generally desired, in other words.

3.7 Conclusion

In this work, we conducted the first study of a new product retrieval task, *i.e.*, mobile app retrieval. Since no test collection exists yet, we created the first one to evaluate this task. We used this test collection to systematically study the effectiveness of both existing retrieval models and a new model that we proposed to effectively leverage the companion review data with apps. Specifically, in order to combine different vocabulary sets in app descriptions and user reviews and to filter out noise in reviews, we proposed a topic model (AppLDA) that jointly models reviews and app descriptions and extracts aligned topics between

reviews and descriptions. Evaluation results show that (1) BM25 outperforms QL when only descriptions are used while QL and CombQL generally outperforms BM25F when reviews are added, (2) leveraging reviews indeed helps app retrieval, (3) AppLDA significantly outperforms traditional retrieval models, and (4) adding priors in AppLDA helps align topics between app descriptions and user reviews.

There are limitations in this work. Since there is no existing test collection and we do not have access to query logs of users, we obtained the test queries leveraging web forums. We collected queries from forum posts, which ask where to find apps with certain features. However, since we examined a few hundreds of such forum posts, we do not have confidence that the collected test queries are indeed representative of user queries. Also, although the query relevance data in section 3.4.3 show that the queries are not very difficult, there may be a bias towards difficult queries since users upload posts to the forum when it is difficult for them to find the mobile apps. Thus, taking a geometric mean of NDCG measures, in addition to an arithmetic mean, may also be useful to find out which methods are more robust on such queries. Lastly, we collected about 43 thousands mobile apps while there are about 1.3 million apps in Google Play app store. The 43 thousands apps cover most of the downloads occurred in the app store since we collected the most popular apps. However, evaluation on the whole data set is expected to be more accurate.

Our work can be further extended in several ways: (1) collecting representative queries when query log data is not available can be further studied, (2) one can explore other directions such as query expansion and feedback-based models for app retrieval problem, and (3) one can identify other characteristics of mobile apps to design a better retrieval model. Our created test collection is made publicly available and thus enables further study of this new problem.

Chapter 4

Product Recommendation via Inference of Implicit Intent in Social Media Text

4.1 Introduction

In the previous chapter, we discussed how to improve product search accuracy leveraging user reviews in order to assist consumers in discovering products. Consumers form a query that best expresses their immediate needs, and the product search engine retrieves products that are most relevant to the query. While product search engines react to queries containing a consumer's explicit intention, we can also analyze consumer text that contains the consumer's implicit intention. In this way, we do not need to wait until the consumer sends a query with the consumer's explicit intention. Instead, we can analyze social media text that contains consumers' implicit intention, and we can recommend products that match the intention well. Since there are countless user text at social media, we have opportunities to recommend products to numerous users.

With rapid development of Internet, people leave massive amount of their status messages on social media. Everyday, 500 million tweets are left on Twitter[1], 55 million status updates are made on Facebook[2], and 80 million photos (with text descriptions) are shared on Instagram[3]. A myriad of such user-generated text data provided researchers opportunities to analyze them. For example, detecting real-time events [89], sentiment analysis in tweets [1], interestingness of tweets [70], and stock market prediction based on tweets [116] have been studied. More recently, researchers studied commercial intention analysis of tweets (*e.g.*, [33, 21]). In social media, people discuss various topics such as their current status including what they do, how they feel, and where they are. Often, social media users express their intention to purchase a product in an implicit or explicit way. For example, a sentence "I will buy an iPhone" contains the user's explicit intention to buy the product. In another sentence "I lost my cellphone," the user does not explicitly express the intention to buy a cellphone, but we can infer that the user may want to purchase a cellphone (implicit intention). Researchers mainly tried to detect whether a user text contains such commercial intention or not. Since commercial intention analysis can provide a way to link "buyers" and "sellers" in social media,

[1] http://www.internetlivestats.com/twitter-statistics/
[2] https://blog.kissmetrics.com/facebook-statistics/
[3] https://www.instagram.com/press/

the role of such analysis system becomes important. Nonetheless, most previous work focused on detection of the commercial intention, and only few work studied on product recommendation based on commercial intention in social media.

In this work, we study the task of supporting user status queries with implicit intention. We formulate the problem as information retrieval problem where we retrieve mobile apps that satisfy the query, *i.e.*, user status text with implicit intention. Our definition of intention has broader coverage than commercial intention in previous studies does; we do not restrict the intention to be commercial. In order to match implicit intention text with mobile apps, (`i`) we first infer possible user intentions in the query and then (`ii`) rank apps based on their relevance to the inferred intentions. For example, given a user query "I am hungry", (`i`) we infer possible intentions such as "find nearby restaurants" and "browse recipe books", and then (`ii`) we recommend apps that can find nearby restaurants or show recipes. Note that the query with implicit intention is different from traditional ad-hoc search queries, where users explicitly express their intentions. To learn such behavior where a user implicitly expresses intention, we leverage social media. We collect tweets that contain information about what people truly desire when they implicitly express their needs. We then build parallel corpora from the tweets and follow information retrieval approach to infer user intention.

Inference of intention in user status text is an important problem. By analyzing intention in user status, product manufacturers or service providers can provide relevant items or targeted ads to the users based on the predicted intention. Such recommendation can also benefit users since they can find products or services they need without expressing their intention explicitly. While users often update user status with explicit intention, they more frequently reveal their needs implicitly. For example, researchers have found that there are about twice as many tweets with implicit commercial intention than those with explicit commercial intention [33, 21]. Therefore, we focus on analyzing implicit intention and recommendation based on it although analyzing implicit intention is more challenging than analyzing explicit intention [21].

This work makes the following contributions:

1. We introduce and study a novel problem of recommending mobile apps for user status text with implicit intention. To the best of our knowledge, there has been no research work that studied the same problem as ours.

2. We propose a novel probabilistic model to retrieve mobile apps that satisfy implicit intention of users. We first infer user intention using parallel corpora we build from social media, and we measure relevance of mobile apps to the inferred intention in order to rank them. To the best of our knowledge, no research work has leveraged parallel corpora for user intention analysis.

3. Since the task has never been performed in the literature, we create a new test data set for evaluating different models. We employ crowdsourcing to label data with query relevance. The test collection is available at `http://timan.cs.uiuc.edu/downloads.html`.

Since our definition of intention is not limited to commercial, our model can be used in various interesting applications besides personalized recommendation and advertising. For example, it can be used in virtual assistant systems such as Apple's Siri and Microsoft's Cortana to suggest services based on implicit user query. Our model also can help other systems prevent certain events such as suicide and mass shooting by analyzing implicit intention in social media user status.

4.2 Related Work

Understanding search intents behind queries has been a great challenge in information retrieval systems. Traditionally, Web search engines take user queries from a single text input box, which makes the systems analyze search intents from a short (about two terms per query [39]) list of keywords. In order to better understand queries, researchers classified queries into different types [9, 43, 53]. For example, Broder [9] classified queries into three classes: navigational, informational, and transactional. The author showed how search engines could evolve by supporting different search intents. Contextual search [56, 12], which uses a user's contextual information to better capture the user's search intent, is related to our work in that we exploit a user's status text in social media, which contains the user's implicit intention. However, we regard each user status text as a single query that contains a user's (implicit) intention, while contextual search requires contextual information in addition to the query. Our work is also related to content-based recommendation systems [80], but our work differs in that we recommend items based on a user's immediate needs in a query while they typically recommend items based on the user's general interests revealed in a user profile.

Query recommendation (suggestion) has been widely studied to recommend alternative related queries given a query, in order to help users who would repeatedly rephrase their queries. To recommend queries, researchers exploited diverse resources such as query logs [3], anchor text [47], and click-through data [13]. Users then choose one of the recommended queries to refine their previous queries. While query recommendation usually requires such interactive processes, query expansion often does not require them and thus can be regarded as automatic query recommendation [3]. Instead of suggesting alternative queries to users, query expansion tries to reformulate queries to resolve vocabulary problem [26], and it is surveyed well in [14]. Our work extends the query expansion in the sense that we automatically adjust a given implicit query

to an explicit query. However, our work differs from query expansion in that the target query is not supposed to be an "expansion" of the original query but "conversion" of the original query to predict user intention in it. Thus, the original query and the expanded query may carry completely different meaning in our work. Query translation [71] and cross-lingual information retrieval [5] are also related to our work in that our work can be seen as translating an implicit query to an explicit query. However, our work differs from them in that the same language is used for the original query and the translated query.

Besides general query intent, researchers studied commercial intention of users in online text, where commercial intention is defined in [16] as "a user has intention to purchase or participate in commercial services." Dai *et al.* [16] defined online commercial intention as commercial intention behind a user's online activities. They developed machine learning models to detect whether a query or Web pages a user visits lead to commercial activity or not. Ashkan *et al.* [2] defined a commercial query as a query with the underlying intention to make a purchase of a product and a noncommercial query as all other queries. Exploiting ad clickthrough logs, query specific information, and the content of search engine result pages (SERP), they classified whether a query is commercial or noncommercial, using a machine learning method. They also predicted ad clickthrough for queries leveraging query intent and the number of displayed ads. Guo and Agichtein [30] exploited fine-grained user interactions such as mouse movements and scrolling so as to infer search intent. Such interactions are converted into features to be used as input to classification models, which classify if a user has research or purchase intent. Our work is related to commercial intent analysis in that we try to connect queries to products (mobile apps). However, unlike those existing studies, we do not classify queries into commercial or noncommercial. Instead, we assume a user needs something, and we recommend products that can satisfy the user needs by inferring user intention from the user's status text.

Recently, commercial intention analysis on social media has attracted researchers' attention. Hollerit *et al.* [33] performed the first commercial intention detection on social media in order to link product buyers and sellers. They distinguished between explicit and implicit commercial intention and stated that implicit intention also has an economic value. They first manually classified tweets into commercial or noncommercial and further classified commercial tweets into explicit or implicit commercial tweets. Then, they proposed an automatic method to detect whether a tweet contains commercial intention or not. Yang and Li [107] also studied commercial intention classification from social media. They developed psychometric measures of user needs through a crowd-sourced study. Then, they built models to predict user needs based on user text. Wang *et al.* [102] classified tweets into six intent categories and one non-intent category. They proposed a semi-supervised learning algorithm for intent classification. Ding *et al.* [21] also exploited social media to identify whether a user text has a commercial intention or not. They found that 625 out of 1,000 commercial

intention tweets contain implicit commercial intention, and they claimed that detecting implicit commercial intention is more challenging. Likewise, most of the studies about social medial commercial intention analysis focused on intention detection in social media. Our work is related to social media commercial intention analysis since we too analyze intention in social media. However, our goal is to further retrieve mobile apps that meet user intention, which we infer from the user text.

Previously, few researchers studied product retrieval problem based on commercial intention tweets. Duan *et al.* [23] mined intention-related products in online question and answer (Q&A) community data. They used a pattern-based method to extract candidate products from the answers. Then, using a collocation extraction model, they measured relevance between intention and products to mine intention-related products. Their work can be regarded as the most similar one to ours since they connected intention with the products. However, our work is different from their work in the following aspects. We focus on how implicit queries can be converted to explicit queries while they do not study such relationship between implicit and explicit queries. Also, they mine products in the Q&A data, so they cannot recommend products that are not present in the data. We do not require products present in the parallel corpora, so the parallel corpora can be used across different domains.

4.3 Problem Definition

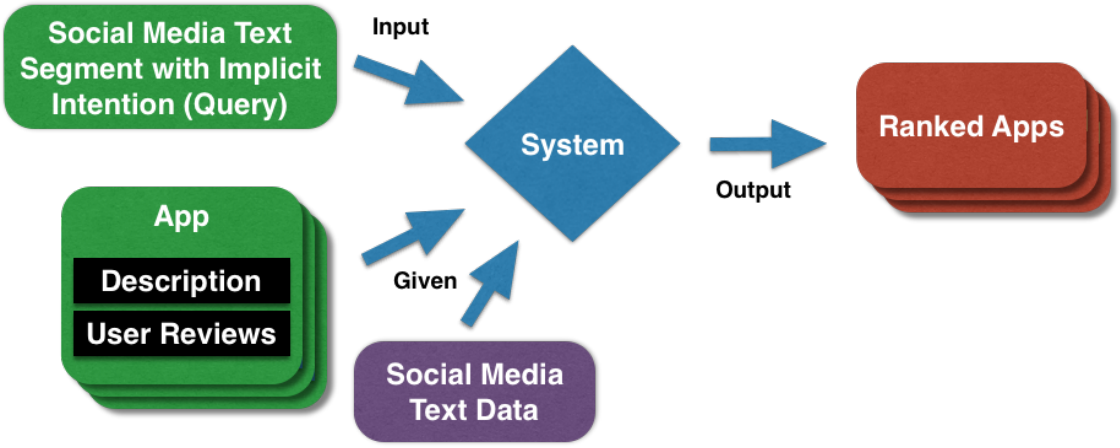


Figure 4.1: Mobile app recommendation for social media text with implicit intention.

In order to recommend mobile apps that would satisfy a user's immediate need that is implicitly expressed in "user status text", we study how to infer a user's intention based on the user's status text segment. This is a new entity retrieval task where the query q is a user's status text segment that implicitly expresses the user need and the entities are mobile apps $\boldsymbol{A} = \{a_1, ..., a_M\}$. We assume q always includes a user intent that

is implicitly expressed. Each mobile app a has its text that represents the app. We are also given social media text data $\boldsymbol{D}$, which we can leverage to infer the user intent. $\boldsymbol{D}$ consists of implicit intention texts $\boldsymbol{D}^{imp}$ and its corresponding explicit intention texts $\boldsymbol{D}^{exp}$. Thus, our goal is to retrieve a list of mobile apps for each q based on their text representations, where the apps are ordered according to their relevance to the user intent in q. We decide to retrieve mobile apps since they have diverse categories that can satisfy various user needs. Also, mobile apps can be downloaded or used in a user's hand so that the user need can immediately be satisfied. Figure 4.1 illustrates our novel problem.

We distinguish between explicit queries and implicit queries. While explicit queries contain their intention clearly, implicit queries do not. For example, user status texts such as "I'm hungry", "I'm so tired", and "I have no one to talk to" do not reveal its intention explicitly while they certainly imply the need for something, and we classify them into implicit queries. The corresponding explicit queries may be "I want food", "relaxing music", and "dating app", respectively, and these types of explicit queries have been used in traditional search engines. In this work, we focus on entity retrieval based on the implicit queries since such queries occur more often in social media text [33, 21].

To the best of our knowledge, retrieval of mobile apps given user status text with implicit intent has not been addressed in previous work. Park *et al.* [78] studied mobile app retrieval problem given an explicit query. While they do not consider implicit queries, we focus on the implicit queries. Baeza-Yates *et al.* [4] studied to predict next mobile app that a user would use based on the user's spatio-temporal contexts. Thus, their work does not require user text but the user's spatio-temporal information, which is different from our work.

Retrieving mobile apps given a user status text with implicit intention is challenging. First of all, social media text is notoriously noisy [44]. For example, there are lots of spam messages on Twitter, and users often ignore grammar when they update their status (tweets). Twitter limits user status text to 140 characters at each time, so it is harder to automatically understand tweets than Web documents, which are usually much longer. Second, a user status q may not have enough similar text in our parallel corpora, which we build to infer user intent. Furthermore, even though there exist enough similar user status texts in the corpora, the hidden intent may be different depending on the users. For example, when a query is "i am hungry", some people might mean "i want a recipe book" while others might mean "i want to find nearby restaurants." Therefore, we need to carefully approach the problem.

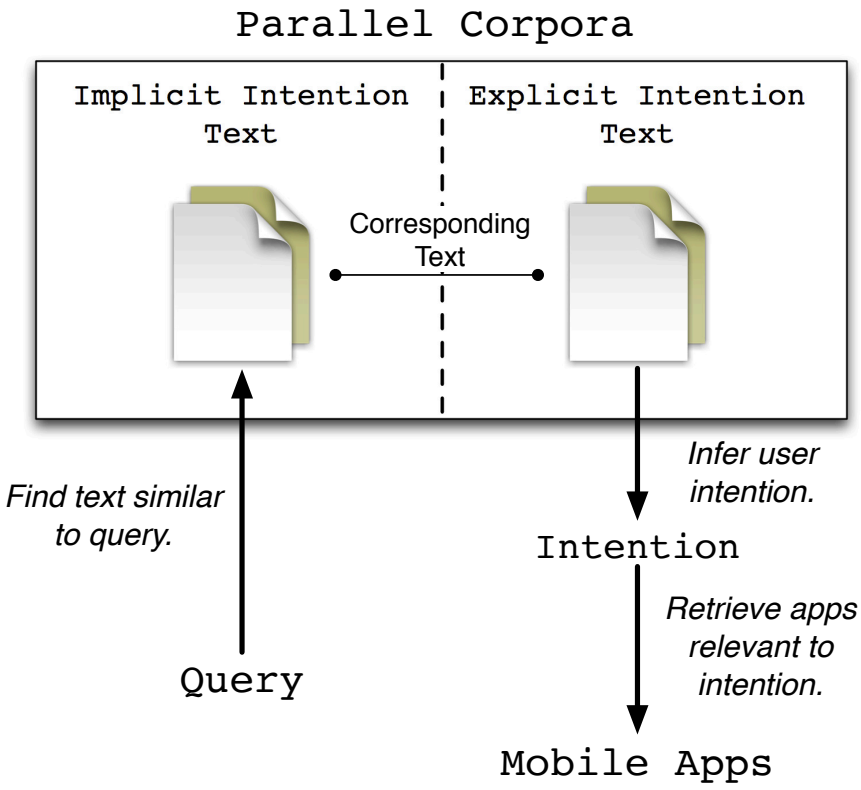


Figure 4.2: Overview of our approach.

4.4 Methods

In order to retrieve mobile apps relevant to user status text with implicit intention, we leverage social media to build parallel corpora, which contain implicit intention text and its corresponding explicit intention text. The overview of our approach is depicted in Figure 4.2. When a user implicitly expresses user needs in a user status text q, we search for similar text in the implicit intention text in the parallel corpora. Then, we infer user intention from the explicit intention text that corresponds to the implicit intention text. Finally, we retrieve mobile apps that are relevant to the inferred intention.

We follow language modeling approach to perform the mobile app retrieval task. In this work, we focus on how to expand the original query language model to capture user intention in the user status text. Thus, our goal is to estimate the query language model $p(w|q)$, which is defined as

$$p(w|q) = (1-\gamma)p_{ml}(w|q) + \gamma p(w|I_q) \tag{4.1}$$

where w is a word, $p_{ml}(w|q)$ is a maximum likelihood estimation of the word w being in the query, and $p(w|I_q)$ is the intention language model for q whose weight is γ. $p_{ml}(w|q)$ is computed by $\frac{count(w,q)}{\sum_{w'} count(w',q)}$ where $count(w,q)$ is the number of w's occurrences in q. We define γ as the confidence on our intention language model. That is, if we are not confident on the estimated intention language model, we give more weight on $p_{ml}(w|q)$, meaning that we do not trust the intention language model and instead trust the original query language model more. Although we build parallel corpora to "translate" implicit intention text into explicit intention text, the inferred intention may not be correct for various reasons, which are discussed in Section 4.3. In such situations, γ can help us adjust our confidence level on the intention language model.

Once we estimate the query language model $p(w|q)$, we retrieve mobile apps relevant to the language model with KL-divergence retrieval model with Dirichlet prior smoothing as presented in [114]. The model is believed as one of the state-of-the-art ad-hoc retrieval models and it can naturally adopt query language model, so we choose it to retrieve mobile apps. The score function to score an app a with respect to q is thus defined as

$$score(a,q) = \left[\sum_{w \in a} p(w|q) \log \frac{p_s(w|a)}{\delta_a p(w|\boldsymbol{A})}\right] + \log \delta_a \tag{4.2}$$

where a word w's smoothed probability $p_s(w|a)$, a background language model $p(w|\boldsymbol{A})$, and the coefficient δ_a are defined as

$$\begin{aligned} p_s(w|a) &= \frac{|a|}{|a|+\tau} \cdot \frac{count(w,a)}{|a|} + \frac{\tau}{|a|+\tau} \cdot p(w|\boldsymbol{A}) \\ p(w|\boldsymbol{A}) &= \frac{count(w,\boldsymbol{A})}{\sum_{w'} count(w',\boldsymbol{A})} \\ \delta_a &= \frac{\tau}{\sum_w count(w,a) + \tau} \end{aligned} \tag{4.3}$$

where $|a|$ is the length of a's text representation, and τ is the Dirichlet prior smoothing parameter. By retaining only the highest probability words in the query language model and re-normalizing it, it can process a query very efficiently [113] with inverted index since only apps containing a query language model word are considered in the formula (4.2). We keep the top 50 words with highest probabilities in the query language model and re-normalize the probability distribution as in the literature [94]. In the rest of this section, we will show the steps of estimating the intention language model $p(w|I_q)$.

4.4.1 Building Parallel Corpora From Social Media

Measuring relevance directly between a query and mobile apps may not be ideal when the query does not explicitly reveal the user intent. We need to understand what the users truly want by writing their status text. In order to infer user intention hidden in the status text, we leverage parallel corpora that contain

texts with implicit intent and their corresponding texts with explicit intent. Thus, a user query with implicit intent is matched with similar text in the parallel corpora to provide us their corresponding explicit intent texts.

To build such parallel corpora, we employ text data at social media. There exist other useful resources to build parallel corpora such as chat logs, movie scripts, and question and answering data. However, chat logs are not generally accessible to public, movie scripts are limited in their amounts, and question and answering data focus more on general knowledge instead of people's intention. Meanwhile, a myriad of user status texts are updated at social media. Not all of them contain people's need, but even small portion of all user status texts are plentiful. In addition, due to the nature of social media such as Twitter, the user status texts are accessible to public. Moreover, because people often leave their "current status" through social media, their contents can be applied to the task well since we want to infer a user's immediate need. Therefore, we choose to employ social media text data to build parallel corpora although they are often very noisy.

Since social media texts are noisy, we employ relatively strict rules to build parallel corpora. In order to match implicit intention texts with explicit intention texts, we make templates such as "i want <EXP> because <IMP>" where <EXP> is explicit intention text and <IMP> is implicit intention text. For example, a user status text "i want to eat pizza because i am hungry" matches our template with the implicit intention text being "i am hungry" and the corresponding explicit intention text being "to eat pizza". In the template, we also use other words than "want" such as "need", "should", and "wanna". Finally, the texts in <EXP> and <IMP> of each user status text become documents in $\boldsymbol{D}^{exp}$ and $\boldsymbol{D}^{imp}$, respectively, while their association is preserved. To filter out noisy sentences, we force restrictions to the matching sentences. For example, we don't let punctuation comes between <EXP> and <IMP>, and we always require sentences starts with "i". There may be other templates that can match implicit intention texts with explicit intention texts. However, to keep high precision instead of high recall, we persist in the templates that are simple yet relatively precise.

Since we match templates with the social media text, building parallel corpora can be efficiently done and linearly scalable with respect to the size of the social media text. In addition, it can be done offline, so this pre-processing step is a one-time effort.

4.4.2 IR Approach to Find Similar Implicit Intention Text

We follow information retrieval approach to match user status text with its similar implicit intention texts in parallel corpora. We employ Query Likelihood retrieval model [84] with Dirichlet prior smoothing [114],

which scores an implicit intent document d^{imp} with respect to q with formulas:

$$\begin{aligned} score(q, d^{imp}) &= \sum_{w \in d^{imp}} count(w, q) \log p(w|d^{imp}) \\ &\propto \sum_{w \in q \cap d^{imp}} count(w, q) \log \frac{p_s(w|d^{imp})}{\delta_{d^{imp}} p(w|\boldsymbol{D}^{imp})} + \log \delta_{d^{imp}} \\ p_s(w|d^{imp}) &= \frac{|d^{imp}|}{|d^{imp}| + \omega} p_{ml}(w|d^{imp}) + \frac{\omega}{|d^{imp}| + \omega} p(w|\boldsymbol{D}^{imp}) \\ p(w|\boldsymbol{D}^{imp}) &= \frac{count(w, \boldsymbol{D}^{imp})}{\sum_{w'} count(w', \boldsymbol{D}^{imp})} \end{aligned} \tag{4.4}$$

We employ this model since it is one of the standard models, and it is relatively easy to tune the parameter. In addition, this model can process a query very efficiently (as efficient as vector space models) with the inverted index. The retrieved documents are sorted by their scores, and we keep top F documents, yielding $\boldsymbol{D}_q^{imp}$, which are regarded as relevant to q. Then, the corresponding explicit intent documents $\boldsymbol{D}_q^{exp} = \{d_1^{exp}, ..., d_F^{exp}\}$ are used to infer user intent in the next step. Please note that this process is similar to pseudo relevance feedback approaches in the literature, but it differs in that it eventually uses the corresponding explicit intent documents instead of the retrieved documents.

4.4.3 Intention Inference with Intention Topic Modeling

In order to infer the intention language model $p(w|I_q)$ from $\boldsymbol{D}_q^{exp}$, we propose to employ intention topic modeling. That is, we first model various user intentions in $\boldsymbol{D}^{exp}$ as pre-processing, and then, we infer the intentions in q using the intention topic models. Employing such intention topic modeling gives us several benefits. First, we can understand various intentions in a given query. The same query may have different intentions depending on the user context, and the intention topics can help us understand such different intentions by inferring multiple intentions. Second, we can remove noisy topics, which impair our intention language model because social media text data are inevitably very noisy. We can exclude intention topics that do not occur enough in $\boldsymbol{D}_q^{exp}$, which are likely noisy topics. Third, the inferred user intentions can be understood by humans in a straightforward way. If the resulting intention language model contains multiple intentions or much noise, humans may not understand the language model well. However, by inferring intention topics and removing noise in $\boldsymbol{D}_q^{exp}$, intention topic modeling can provide a list of intentions in q, which humans can understand more easily.

We first pre-process $\boldsymbol{D}^{exp}$ to model user intentions by topic modeling approach, Latent Dirichlet Allocation (LDA) [8], with parameters α, β, and K. The learned word probability distribution for each intention topic $\hat{\boldsymbol{\phi}}_k$ is then given to our Intention LDA in order to infer intentions in $\boldsymbol{D}_q^{exp}$. The graphical representation of Intention LDA is depicted in Figure 4.3. The generative story of the explicit intention text in $\boldsymbol{D}_q^{exp}$ is as

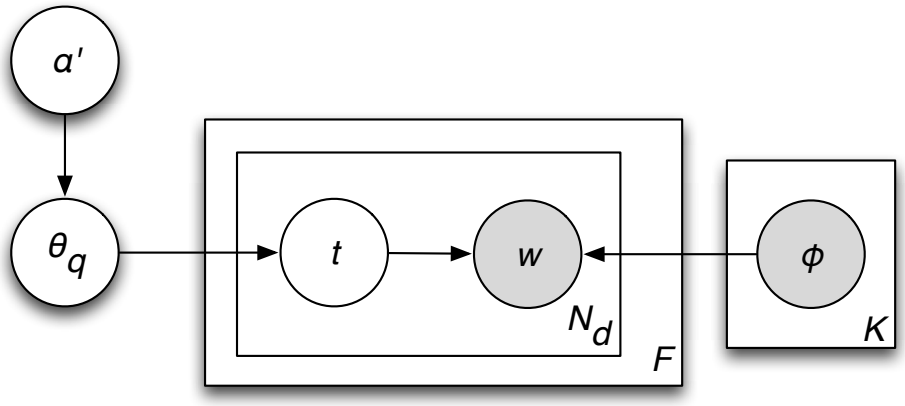


Figure 4.3: Graphical representation of Intention LDA. F is the number of explicit intent documents for a query, N_d is the number of words in each document, and K is the number of intention topics.

follows. For each word of explicit intention text d, the author first chooses an intention $t_{d,i}$ according to the query-level intention distribution $\boldsymbol{\theta}_q$, which is drawn from a Dirichlet distribution with a symmetric vector $\boldsymbol{\alpha'}$. Then, the author chooses a word $w_{d,i}$ according to a word distribution $\hat{\boldsymbol{\phi}}_{t_{d,i}}$, which is pre-estimated from $\boldsymbol{D}^{exp}$ by LDA. This process is repeated for all words in $\boldsymbol{D}_q^{exp}$ for I iterations.

The collapsed Gibbs sampling formula to infer the intention assignment for each word $t_{d,i}$ is defined as

$$\begin{aligned} p(t_{d,i}|\hat{\boldsymbol{\Phi}}, \boldsymbol{T}_{\backslash d,i}, \boldsymbol{\alpha'}) &\propto p(w_{d,i}|t_{d,i}, \hat{\boldsymbol{\Phi}}) \cdot p(t_{d,i}|\boldsymbol{T}_{\backslash d,i}, \boldsymbol{\alpha'}) \\ &\propto p(w_{d,i}|\hat{\boldsymbol{\Phi}}_{t_{d,i}}) \cdot \frac{N^{\backslash d,i}_{t_{d,i}|\boldsymbol{D}_q^{exp}} + \alpha'}{N_{\boldsymbol{D}_q^{exp}} - 1 + K\alpha'} \end{aligned} \tag{4.5}$$

where $\boldsymbol{T}$ is a set of all intention assignments in $\boldsymbol{D}_q^{exp}$, "$\backslash d, i$" means excluding d's ith data, and $\boldsymbol{\alpha'}$ is a symmetric vector $\{\alpha', ..., \alpha'\}$. N is the number of words that satisfy the superscript and subscript conditions. Thus, $N^{\backslash d,i}_{t_{d,i}|\boldsymbol{D}_q^{exp}}$ is the number of words in $\boldsymbol{D}_q^{exp}$ assigned with the intention $t_{d,i}$ excluding the ith word in d, and $N_{\boldsymbol{D}_q^{exp}}$ is the number of all words in $\boldsymbol{D}_q^{exp}$.

This model is similar to the regular LDA when it is applied to infer topics in new documents, except that the intention distribution $\boldsymbol{\theta_q}$ is estimated at a query-level, not a document-level. This is because our data set is expected to be very sparse. Both implicit and explicit texts consist of a few words, so our documents are much shorter than the typical documents. This problem of short text results in less word co-occurrence data, so the topics cannot be estimated well. As addressed in [34], to solve the problem, one can aggregate short texts to make a large text, if the short texts are believed to have similar topic distributions; this approach will have the same effect in the end. Similarly, since we can assume that the retrieved texts in $\boldsymbol{D}_q^{exp}$ for q have the similar topic distributions, we use the query-level estimation for $\boldsymbol{\theta_q}$.

Building Intention Language Model Now, we can build the intention language model $p(w|I_q)$ with the estimations from the intention topic model.

$$p(w|I_q) = \sum_{t \in \text{Intentions}(q)} p(w|t, \boldsymbol{D}_q^{exp}) \cdot p(t|\boldsymbol{D}_q^{exp}) \tag{4.6}$$

where Intentions(q) is a set of candidate intentions for q. In order to remove noisy topics as discussed earlier in this section, we do not utilize all possible topics. We instead keep only top X intentions according to $p(t|\boldsymbol{D}_q^{exp})$, which is the probability of the intention being in the retrieved explicit intention text. Intuitively, a few top intentions for q can satisfy general users' needs, we exclude the other intentions that are likely to be noisy. The probabilities $p(w|t, \boldsymbol{D}_q^{exp})$ and $p(t|\boldsymbol{D}_q^{exp})$ are defined as

$$\begin{aligned} p(w|t, \boldsymbol{D}_q^{exp}) =& \frac{\hat{N}_{t|\boldsymbol{D}_q^{exp}}}{\hat{N}_{t|\boldsymbol{D}_q^{exp}} + \mu} \cdot p_{ml}(w|t, \boldsymbol{D}_q^{exp}) + \frac{\mu}{\hat{N}_{t|\boldsymbol{D}_q^{exp}} + \mu} \cdot p(w|\hat{\boldsymbol{\Phi}}_t) \\ p(t|\boldsymbol{D}_q^{exp}) =& \frac{\hat{N}_{t|\boldsymbol{D}_q^{exp}} + \alpha'}{\hat{N}_{\boldsymbol{D}_q^{exp}} + K\alpha'} \end{aligned} \tag{4.7}$$

where $\hat{N}$ and $\hat{\boldsymbol{\Phi}}_t$ are estimations from the intention topic model and the regular LDA, respectively, μ is the smoothing parameter. $p(t|\boldsymbol{D}_q^{exp})$ is normalized to have its sum equal to one if $\boldsymbol{T_q}$ doesn't contain all possible intentions. Here, we smooth $p(w|I_q)$ in a topic-level. That is, for each topic, $p(w|t, \boldsymbol{D}_q^{exp})$ is smoothed with its posterior estimation from LDA by Dirichlet prior smoothing. Thus, its ML estimation, $p_{ml}(w|t, \boldsymbol{D}_q^{exp}) = \frac{\hat{N}_{w,t|\boldsymbol{D}_q^{exp}}}{\hat{N}_{t|\boldsymbol{D}_q^{exp}}}$, gets higher weight if more words are assigned with t. When $\mu = 0$, the estimation is the same as maximum likelihood estimation, and when μ approaches to infinity, the estimation becomes the same as posterior estimation from LDA. With such topic-level smoothing, we can expect two major benefits. Firstly, we can dynamically smooth the intention topics depending on the number of assigned words. Intuitively, if more words are assigned with a topic t, we can more trust its ML estimation since we have more evidence about t in the text $\boldsymbol{D}_q^{exp}$. In addition, we can "customize" each intention topic model with q. When a user status text q contains intentions that are not captured well by existing intention topics, the per-topic smoothing can help transform existing intention topics into new intention topics with the evidence present in the retrieved text. Please note that, in order to obtain reliable probability distributions, we run multiple Markov chains, and take the average values of the topic assignments.

Computational Complexity The complexity of the pre-processing steps, which is modeling user intentions with collapsed Gibbs sampling from $\boldsymbol{D}^{exp}$, is $O(I \cdot W \cdot K)$ where I is the number of iterations, W is the number of all words in $\boldsymbol{D}^{exp}$, and K is the number of intention topics. Therefore, it is linearly scalable

with respect to the size of the social media text data ($\boldsymbol{D}^{exp}$). This pre-processing can be done offline, thus a one-time effort.

On the other hand, user intention inference is done online. With collapsed Gibbs sampling, the complexity is $O(I' \cdot W_q \cdot K)$ where I' is the number of iterations for inference, and W_q is the number of all words in $\boldsymbol{D}_q^{exp}$. The complexity of building intention language model is $O(K \log K + V \cdot T)$ including choosing top T intentions , $(O(K \log K))$, and computing word distribution for T intentions, $(V \cdot T)$, where V is the vocabulary size. In general, I' is small (*e.g.*, 100), and W_q is small (*e.g.*, 875 words on average) since $\boldsymbol{D}_q^{exp}$ does not depend on the size of the whole $\boldsymbol{D}^{exp}$. Also, $T \leq K$, so inferring user intention and building intention language model has an average complexity of $O(K) + O(K \log K + V \cdot T) = O(K(\log K + V))$. The vocabulary size V sublinearly increases as the size of data increases, so this process is sublinearly scalable with respect to the size of data. As discussed in multiple places of this section, other components of the system are also scalable and perform efficiently, so the whole process is scalable and performed efficiently.

4.5 Experimental Setup

4.5.1 Data Set

In order to perform experiments for the task, we need data sets such as mobile apps, parallel corpora from social media, and the test queries and their relevance data. We use mobile app data in [78], which were crawled from Google Play App Store[4]. We omitted apps in game categories, which takes about 38% of the whole data, since our goal is to recommend more practical solutions while games are for entertainment in general. The resulting data set contains 26,832 apps in total. Since user status text is often written in an informal way by people in general, we also use apps' user review texts, which are also often written in an informal way. On average, there are 31.4 reviews for each app. The experiment results in this work will be based on concatenated text of app descriptions and user reviews unless otherwise specified. We tokenized text into word tokens and lemmatized them using Stanford CoreNLP [66] version 1.3.5. We lowered all word tokens and removed punctuation, stopwords, and word tokens that appear in less than five app descriptions (and five user reviews if reviews are also used). More statistics of the resulting data are shown in Table 4.1. Note that about 30% of vocabulary is omitted when we add user reviews; this is because many words in app descriptions are indeed not used in user reviews.

Since we study a new task that has not been studied before, there is no existing test collection or parallel corpora available to use. In the rest of this section, we describe how we collect such data in detail.

[4] `https://play.google.com/store/apps`

Table 4.1: Statistics of text data in apps without reviews and with reviews.

	without reviews ($\neg \boldsymbol{R}$)	with reviews ($\boldsymbol{R}$)
Average number of tokens	116.5	395.6
Total number of tokens	3,124,611	10,615,767
Vocabulary size	20,293	14,196

Collecting Tweets to Build Parallel Corpora

We already described how to build parallel corpora with social media text data in Section 4.4.1. Here, we describe how to collect tweets and the details of the data set. Among various social media, we decided to crawl the data from Twitter mainly because plenty of tweets are available to public, which means that more user status data would match our templates. We searched for tweets containing our keywords using Twitter's search function. The query we used is, for example, *"i want" because until:YYYY-MM-DD*, which is supposed to search for tweets containing the word *because* and the exact phrase *i want* until the specified date. We crawled tweets dated between June 6, 2006 and November 4, 2015. Then, we removed tweets that do not satisfy the templates in Section 4.4.1, and cleaned the texts in the same way as the app text data are cleaned. We excluded tweets that do not have any word tokens in implicit or explicit texts. We also allowed only one (implicit text, explicit text) pair if there exist exactly the same pairs, to avoid noise. Finally, we obtained 1,609,894 (implicit text, explicit text) pairs from 1,115,948 unique Twitter users where implicit text contains 2.7 word tokens and explicit text contains 2.5 word tokens on average, and its vocabulary contains 35,695 unique words.

Collecting Representative Queries

We need to obtain representative queries so as to evaluate whether the proposed model is indeed useful. Unfortunately, such test queries and their relevance data do not exist yet, so we create our own test collection. It is, however, challenging to obtain "representative" user status text segments. We cannot simply rank tweet texts based on the number of tweets that contain exactly the same content or similar content in the whole tweet data set; people use different vocabulary to describe similar statuses, and the high-ranked tweet texts may have similar themes. Thus, we employ topic model to first find main themes in tweets and choose a representative tweet from each theme. We crawled tweets containing a keyword "i" so that the retrieved tweets are likely to be about the user's own status. The crawled tweets are dated between June 20, 2015 and August 18, 2015, and after cleaning them, we obtained 960,874 tweets. With LDA, we modeled 50 topics and ranked tweets with KL-divergence scoring function in formula (4.2) for each topic. After removing nine topics that are mostly spams, we asked a domain expert to extract the first user status segment that implicitly

expresses user need, starting from the most relevant tweets, for each of the 41 topics.[5] We recognize that implicitly expressed user needs are related to the user's mood, so we added mood words from [104], where emotions words are assigned for each of eight emotions. To choose 12 mood words from them, we omitted the words that do not overlap with the existing queries and that occur less than 400 times in the tweets we crawled. Then, we made a query with those words by the template "i am <word>" or "i feel <word>", whichever occurs more according to Google's exact phrase search. After removing nine queries that are discussed in the next section, we compiled 44 queries, each of which contains around 5.5 words on average.

Collecting Query Relevance Data

Since labeling all the retrieved mobile apps from various models for each query is too expensive, we created a pool. In specific, for each query, we pooled together the top 20 retrieved apps from each of multiple retrieval models with various parameter settings, in order to acquire enough apps that are most likely relevant. We then employed a crowd-sourcing service, CrowdFlower[6], to label the (query, app) pairs at affordable price. Each worker was asked to read a query and a corresponding app's name and description, and follow the link to the app store page if needed. Then, the worker was paid three cents to judge if the app satisfies the user need on three relevance levels (no satisfaction at all (0), makes sense with some context (1), and perfect satisfaction (2)).

To ensure the quality of judgments, we let each (query, app) pair be judged by three annotators, and we used "quiz" function[7] in CrowdFlower to remove users who do not score high enough on the quizzes we made. The three resulting judgments for each (query, app) pair were averaged to be used as a relevance score. We removed nine queries that retrieved less than five perfectly relevant apps (by majority vote) since they can harm the reliability of evaluation results.

In total, 156 workers were employed through the crowd-sourcing service, and each of them made 177.0 $\pm$ 117.5 judgments where the number after $\pm$ is standard deviation. After all, we obtained 180 $\pm$ 59.5 aggregated relevance judgments on average for each query. We measured the inter-annotator agreement by Fleiss' kappa, which was 0.31, where the value between 0.21 and 0.4 can be interpreted as fair agreement according to [50].

[5] Note that the domain expert had to go through many tweets for some topics that contain only few implicit intentions.
[6] `http://www.crowdflower.com/`
[7] A worker's judgments are stored only if the worker passes with accuracy above 75% from the initial quiz with eight questions and maintains the accuracy above 75% while the worker is given one random quiz question for every three regular questions in random order.

4.5.2 Evaluation Metrics

The relevance judgments for each (query, app) pair by three annotators are averaged, and the aggregated real value ranges between 0 and 2. Thus, we employ Normalized Discounted Cumulative Gain (NDCG) [40] as the evaluation metric since it can handle multiple-level relevance data while metrics such as Mean Average Precision cannot. We measure NDCG at 3, 5, 10, and 20 top retrieved apps to reflect information needs of various users. Note that NDCG@3 is more important for this task than for traditional Web search since only a few mobile apps can fit a mobile phone screen well, and many social media users use their mobile phones to connect to social media.

4.5.3 Baseline Methods

We have two types of baseline methods: models that do not leverage parallel corpora and models that do. The models not leveraging parallel corpora include Query Likelihood Language Model (**QL**) and **Relevance** model [52], which are standard information retrieval models based on language model. We use Query Likelihood Language Model with Dirichlet prior smoothing as described in Section 4.4.2 where we score an app a instead of d^{imp}. Since our Intention model expands the original query with online processing, we use Relevance model (Model 1) to be fair, which is one of the state-of-the-art query expansion methods with online processing. The Relevance model first retrieves apps with original query using QL. Then, it expands the original query with the descriptions of the top F_a retrieved apps.

The models that leverage parallel corpora are closely related to cross-lingual information retrieval models [5] because parallel corpora are usually exploited for two different languages. We employ **Translation** model and Cross-Lingual Relevance Model (**CL-Relevance**) [51]. Translation model employs IBM Model 1 [10] to estimate word translation probabilities from $\boldsymbol{D}^{imp}$ to $\boldsymbol{D}^{exp}$. Then, it builds the query language model by

$$p(w|q) = \sum_{w' \in V(\boldsymbol{D}^{imp})} tr(w|w')p(w'|q) \tag{4.8}$$

where $V(\boldsymbol{D}^{imp})$ is a vocabulary set in $\boldsymbol{D}^{imp}$, and $tr(w|w')$ is the translation probability from w' to w. Thus, the resulting query language model would consist of words in $\boldsymbol{D}^{exp}$ that are semantically associated with the original query. CL-Relevance is an extension of the Relevance model and is one of the state-of-the-art cross-lingual information retrieval models. Similar to our **Intention** model, CL-Relevance leverages the parallel corpora to retrieve similar intention texts from $\boldsymbol{D}^{\boldsymbol{exp}}$ given a query as in Section 4.4.2, and then, it estimates the query language model from the top F intention texts.

Translation and CL-Relevance models are designed for cross-lingual information retrieval. However, our

task involves only one language although we built the parallel corpora with $\boldsymbol{D}^{imp}$ and $\boldsymbol{D}^{exp}$, which means that we can incorporate the original query into the estimated query language model since they are made of the same language. We interpolate the original query language model with the estimated query language model with a fixed coefficient γ as in (4.1), where the estimated query language model from Translation or CL-Relevance becomes the intention language model. The resulting models for Translation and CL-Relevance models are called **Translation+ORIG** and **CL-Relevance+ORIG**, respectively.

4.5.4 Parameter Setting

We tuned the parameters with a data set containing both app descriptions and user reviews, and we use the data set unless otherwise specified. The parameter values are shown in Table 4.2, and we use those values in this work unless otherwise specified. To model topics in $\boldsymbol{D}_q^{exp}$ with LDA for Intention model, we run 1,000 Gibbs sampling iterations. We use three Markov chains with 100 Gibbs sampling iterations each for Intention LDA. We use GIZA++ package [72] for Translation model, where we use IBM Model 1 with default parameter values.

Table 4.2: Parameter setting for QL (Q), Relevance (R), Translation (T), Translation+ORIG (TO), CL-Relevance (C), CL-Relevance+ORIG (CO), and Intention (I) models.

	Value	Models
γ	0.9(TO), 0.8(CO, I)	TO, CO, I
τ	1,000	Q, R, T, TO, C, CO, I
ω	100	C, CO, I
F	350	C, CO, I
F_a	50	R
λ	0.8(R), 0.5(C, CO)	R, C, CO
α	0.01	I
β	0.01	I
K	300	I
α'	0.1	I
X	5	I
μ	5	I

4.6 Result Analysis

4.6.1 Qualitative Analysis

Table 4.3 shows top intention topics obtained from social media user's explicit intention text $\boldsymbol{D}^{exp}$. It seems that the intentions are closely related to people's everyday life such as cleaning room, watching TV shows, eating or making food, playing sports, and supporting sports teams. For each query, we infer the most

Table 4.3: Top five intention topics by LDA. Words in each column represent a topic.

room	show	eat	play	win
clean	watch	make	football	super
fridge	tv	chicken	team	seahawks
tidy	stop	cheese	player	bowl
house	netflix	soup	soccer	raven
living	reality	egg	basketball	bronco
mini	series	fries	game	ravens
closet	ellen	potato	baseball	49ers
lock	movie	bacon	sport	lose
bigger	program	mac	fantasy	simply

Table 4.4: Top query language model words and their probabilities estimated from each model for a query "my phone died". MLE indicates maximum likelihood estimation of the query.

MLE		Translation+ORIG		CL-Relevance+ORIG		Intention	
pretty	0.25	work	0.11	sleep	0.26	sleep	0.34
tire	0.25	pretty	0.08	bed	0.09	bed	0.13
work	0.25	day	0.07	work	0.08	nap	0.06
today	0.25	today	0.07	today	0.07	work	0.06
-	-	tire	0.06	pretty	0.07	today	0.06
-	-	tomorrow	0.05	tire	0.07	tire	0.05
-	-	sleep	0.04	stop	0.06	pretty	0.05
-	-	feel	0.04	play	0.05	plan	0.03
-	-	hour	0.02	friend	0.02	start	0.02
-	-	week	0.02	hour	0.01	back	0.02

Table 4.5: Top five intentions inferred from our Intention model for each query. Each intention is represented by its top five words.

query	i'm getting sad	i feel sleepy	i feel lonely
top intentions	happy, make, feel, smile, sad	sleep, back, wake, day, home	friend, make, internet, talk, guy
	hug, give, kiss, big, nus	coffee, drink, energy, cup, caffeine	buddy, cuddle, friend, texting, text
	listen, stop, song, radio, station	nap, stop, home, school, work	watch, movie, stop, film, cry
	listen, music, stop, play, ipod	sleep, bed, start, earlier, early	follow, fan, account, friend, 5so
	life, people, thing, positive, stop	late, stay, stop, night, sleep	hug, give, kiss, big, nus
query	i got a bad feeling about today	i spend way too much money	i am hungry
top intentions	start, people, listen, advice, hang	shopping, store, grocery, shop, work	eat, start, breakfast, dinner, lunch
	god, pray, jesus, lord, prayer	stop, online, buy, thing, stuff	food, eat, stop, make, bring
	cry, tear, joy, happiness, happy	money, job, give, make, lot	eat, stop, food, hungry, bore
	medicine, sleep, doctor, med, pain	card, credit, gift, buy, give	eat, make, chicken, cheese, soup
	cry, ball, curl, bed, die	job, find, asap, pay, good	pizza, eat, order, work, food

Table 4.6: Top retrieved apps from each model for query "i'm pretty tired after work today".

QL	Translation+ORIG	CL-Relevance+ORIG	Intention
Wheel Tire Calculator	Sleep Cycle	Sleep Cycle alarm clock	Sleep Better with Runtastic
Tires Plus	Sleep Cycle alarm clock	Sleep Diary Pro	Sleep Diary Pro
Pregnancy Workout Today	Sleep Hypnosis: Cure Insomnia	Sounds for Baby Sleep Music	Sleep Cycle
Tire Calculator PRO	Nature sounds to sleep	Sleep Analyzer	Sleep Cycle alarm clock
Pretty Calculator	Horoscopes for Facebook	Sleep Better with Runtastic	Sleep as Android Unlock
ForzaTune 5	Sleep Better with Runtastic	Sleep Cycle	Sounds for Baby Sleep Music
Verizon Roadside Assistance	Squats	Sleep Talk Recorder	Relax Melodies: Sleep & Yoga
Boxing Interval Timer	Fetch: Your Buying Assistant	Sleep as Android	Deep Sleep and Relax Hypnosis
7-Eleven, Inc.	SleepyTime: Bedtime Calculator	SleepBots - Sleep Cycle Alarm	Sleep Analyzer
Shut Up Button	Dog Licker Live Wallpaper FREE	Sleep as Android Unlock	Relax Music & Sleep Cycle

such as "budget manager".

Table 4.4 shows the query language models estimated by different methods for a query "i'm pretty tired after work today." While MLE gives the same probability to each keyword of the query, the other models expand the query language model with our parallel corpora. The language model estimated by Translation+ORIG model does not seem to highlight words that are important specifically for the query. As discussed in the next section, the intention language model estimated by Translation+ORIG model is in more general context. For example, words such as "pretty", "tomorrow", "hour", and "week" do not have to be emphasized much, and words such as "sleep" and "bed" need to be emphasized more for the query. CL-Relevance+ORIG model assigns more weights to such words, and Intention model assigns even more weights to them, so they retrieve more relevant apps. Table 4.6 shows the top retrieved apps for the same query. QL, which assigns the same weight to each query word as MLE does, retrieves many irrelevant mobile apps such as "Wheel Tire Calculator", "Tires Plus", "Pregnancy Workout Today", and so on. Translation+ORIG model retrieves apps such as "Horoscope for Facebook", "Squats", "Fetch: Your Buying Assistant", and so on, which do not seem to satisfy the user intention in the query, and this can be expected from its poorly estimated query language model. Most apps retrieved by CL-Relevance+ORIG and Intention models are satisfying since they could build the query language models well as shown in Table 4.4.

4.6.2 Quantitative Analysis

Table 4.7: NDCG measures for baseline models and our Intention model. QL and Relevance models do not leverage parallel corpora while Intention model does. $\neg\boldsymbol{R}$ does not exploit user reviews of apps while $\boldsymbol{R}$ does. (% improve.) indicates Intention model's percentage improvement over Relevance model. The symbols * and $^\circ$ indicates statistical significance (student's two-tailed paired t-test with $p<0.05$) over QL and Relevance, respectively.

Data	Model	N@3	N@5	N@10	N@20
$\neg\boldsymbol{R}$	QL	0.404	0.420	0.415	0.418
	Relevance	0.437	0.430	0.441	0.461
	Intention	**0.531**$^{*\circ}$	**0.534**$^{*\circ}$	**0.527**$^{*\circ}$	**0.515***
	(% improve.)	+21.5%	+24.2%	+19.5%	+11.7%
$\boldsymbol{R}$	QL	0.445	0.438	0.440	0.438
	Relevance	0.463	0.460	0.444	0.455
	Intention	**0.596**$^{*\circ}$	**0.585**$^{*\circ}$	**0.564**$^{*\circ}$	**0.560**$^{*\circ}$
	(% improve.)	+28.7%	+27.2%	+27.0%	+23.1%

Table 4.7 compares Intention model with models that do not leverage our parallel corpora, and it also shows measures when reviews are used and not used. Relevance model indeed outperforms QL with its query expansion from the app data set. Intention model significantly outperforms QL and Relevance models because it can infer user intention by leveraging the parallel corpora. Intention model improves the other

Table 4.8: NDCG measures for models leveraging parallel corpora. (% improve.) indicates Intention model's percentage improvement over CL-Relevance+ORIG. The symbols * and ° indicates statistical significance (student's two-tailed paired t-test with p<0.05) over Translation+ORIG and CL-Relevance+ORIG, respectively.

Data	Model	N@3	N@5	N@10	N@20
	Translation	0.480	0.486	0.480	0.498
	CL-Relevance	0.376	0.374	0.363	0.386
$\neg \boldsymbol{R}$	Translation+ORIG	0.468	0.472	0.472	0.474
	CL-Relevance+ORIG	0.477	0.470	0.473	0.484
	Intention	**0.531**°	**0.534***°	**0.527***°	**0.515**
	(% improve.)	+11.3%	+13.6%	+11.4%	+6.4%
	Translation	0.483	0.481	0.478	0.482
	CL-Relevance	0.425	0.409	0.395	0.406
$\boldsymbol{R}$	Translation+ORIG	0.519	0.519	0.503	0.510
	CL-Relevance+ORIG	0.526	0.513	0.510	0.507
	Intention	**0.596**°	**0.585**°	**0.564***°	**0.560**°
	(% improve.)	+13.3%	+14.0%	+10.6%	+10.5%

models especially for NDCG at 3 and 5, which is desirable for mobile environment. In general, exploiting user reviews is shown to be beneficial for product search, which is also found in [78]. Interestingly, when reviews are used, Intention model's percentage improvement over Relevance model increases from 19.2% to 26.5% on average. Intuitively, while app descriptions are written by app developers, user reviews and user status texts are written by ordinary people with similar vocabulary, so the effect of exploiting parallel corpora can be amplified by using user reviews.

Table 4.8 compares models leveraging our parallel corpora. Again, it seems that all the models except Translation model improve when reviews are used. CL-Relevance does not perform well because each text pair in our parallel corpora is relatively short and noisy so that the top retrieved text from the corpora may be corrupted with noise. We mix the original query language model and the query language model estimated from CL-Relevance with linear interpolation because our task involves only one language as discussed in Section 4.5.3. When it is mixed with the original query language model as CL-Relevance+ORIG, it improves quite much, which means that the text in parallel corpora can be very helpful if properly used. Translation model outperforms CL-Relevance as itself, which means that query expansion with the whole parallel corpora may be better than that with only the top retrieved texts from the parallel corpora, when the parallel corpora is relatively noisy. However, when Translation model is mixed with the original query language model as Translation+ORIG, it performs comparably with CL-Relevance+ORIG. Our Intention model significantly outperforms all the other models in general. Although Intention model infers user intention from only top retrieved texts from the parallel corpora, it effectively removes the noisy intentions, yielding good results.

In order to filter out noisy intentions, Intention model uses only top X intention topics and estimates

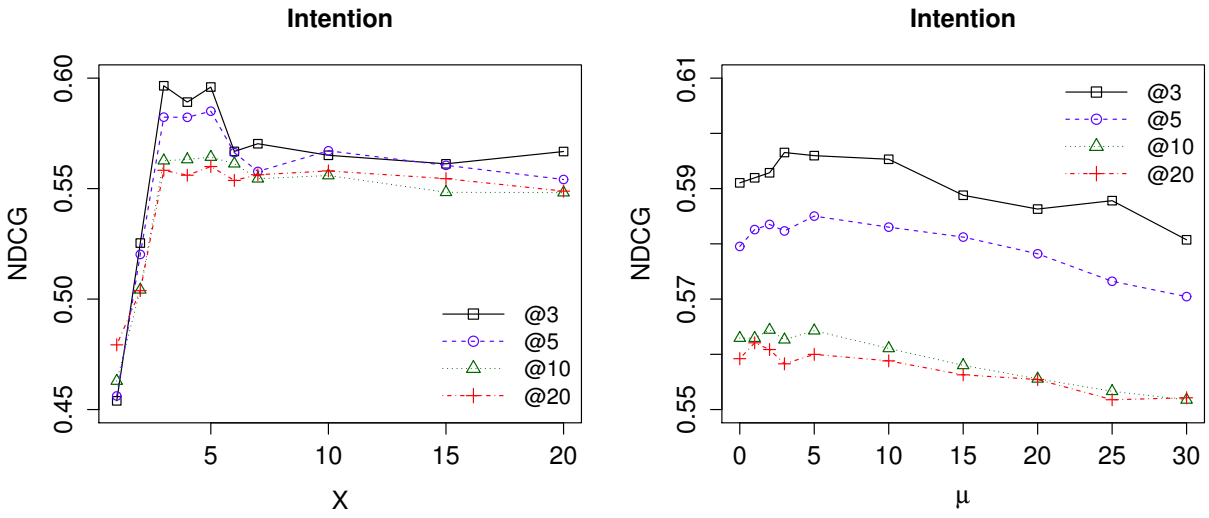


Figure 4.4: NDCG measures for different X (left) and μ (right) values of Intention model.

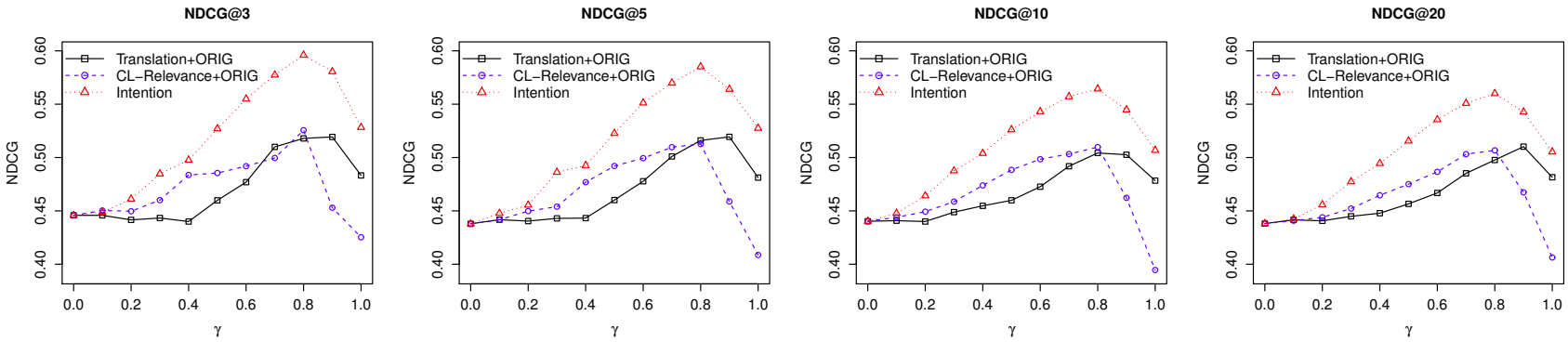


Figure 4.5: NDCG measures for different γ values of models that leverage the parallel corpora.

intention topics with per-topic smoothing. Figure 4.4 depicts Intention model with different number of top intention topics (X) values and different amount of smoothing (μ). The idea of removing unpopular intention topics, which are regarded as noisy topics, seems to yield better results indeed, especially for NDCG at 3 and 5. When we keep only one or two top intentions, Intention model does not perform well. However, its performance is relatively good when X is between 3 and 5, and then the performance degrades as it keeps more intentions. Meanwhile, we can assign more weights to the intention topics pre-estimated from LDA by increasing μ value. The figure shows that adding a small amount (μ between 1 and 5) of the pre-estimated topics helps a little, but adding too much of them worsens the performance. From the results, it seems that removing noisy intentions plays a more important role than exploiting pre-estimated topics in estimating a good intention language model.

Translation+ORIG, CL-Relevance+ORIG, and Intention models, which leverage the parallel corpora, combine the original query language model and the intention language model, estimated from the parallel corpora, with a linear interpolation parameter γ as in equation (4.1). When $\gamma = 0$, the query language model becomes the original query language model $p_{ml}(w|q)$, and when $\gamma = 1$, the query language model is purely the intention language model from the parallel corpora. That is, when $\gamma = 1$, Translation+ORIG and CL-Relevance+ORIG models become Translation and CL-Relevance models, respectively. Figure 4.5 depicts the models with different γ values. We can see that relying more on the intention language model results in better performance in general, which means that leveraging our parallel corpora is indeed important for the task. Intention model seems to outperform the other models with almost all γ values, even when the query language model comes entirely from the parallel corpora ($\gamma = 1$). Interestingly, all the models perform better when the original query language model is incorporated than when it is not. This means that the original query language model and the intention language model have their own strong points, so their combination makes even better results.

4.7 Conclusions and Future Work

In this work, we studied the problem of mobile app retrieval for social media text that contains a user's implicit intention. Recommending mobile apps for such text can be very useful for both users and app developers, but no previous work has addressed this novel problem. We proposed how to build parallel corpora that can convert implicit intent text into explicit intent text, and we proposed the Intention model that leverages the parallel corpora to infer user intent to recommend satisfying mobile apps. In specific, Intention model infers user intention via Intention LDA and builds an intention language model by filtering

noisy intentions out. Evaluation results show that (i) leveraging our parallel corpora built from social media is indeed beneficial for the task, (ii) exploiting user reviews further amplifies the effects of leveraging the parallel corpora, (iii) removing noisy intentions plays an important role in Intention model, (iv) the original query language model and the estimated intention language model of Intention model complement each other well, and (v) intentions inferred from Intention model help us understand various user intentions in a fairly straightforward way.

There are limitations in this work. When we collect the parallel corpora, we exploited templates to match texts with certain patterns. Those templates were manually generated in order to ensure high precision. However, we do not know how generalizable they are for other similar tasks because we did not investigate accuracy of the collected corpora. Also, the templates do not ensure high recall since there may be many more templates we can exploit. When we evaluate the retrieved mobile apps from the suggested methods, we tested them with a relatively small number of queries since it is expensive to obtain query relevance data. More queries along with their relevance data would help us ensure higher validity of the evaluation results.

Our work can be further extended in several ways. First, while we recommended mobile apps, one can recommend other entities such as products of other types or services, leveraging our parallel corpora. Second, one can recommend mobile app pages instead of mobile apps so that a user can open a needed app page right away by analyzing user text. Lastly, since our parallel corpora made from social media are domain independent, one can leverage the corpora to other applications such as chat bots or virtual assistants to understand user intention. Our test collection is released to public so that it enables further study of this new problem.

Part II

Assisting Users in Making Purchase Decisions

Chapter 5

SpecLDA: Modeling Product Reviews and Specifications to Generate Augmented Specifications[1]

5.1 Introduction

In the previous two chapters, we discussed how we can assist users in discovering products. Specifically, we proposed how we can improve search accuracy using user reviews, and how we can recommend products based on social media text that contains implicit intention. Once a user finds a product that matches the user's intention, the user will look into the details of the product or look for other users' opinions so that the user can make a purchase decision. For example, users would first view descriptions or specifications of products in order to know details of the products they found. If the products still match the users' interests, then the users might look for other users' opinions on the products. Then, the user would make a purchase decision; if the user still likes the product, the user would purchase the product, but if not, the user would look for another product. In this chapter, we generate augmented specifications to help the users understand specifications so that they can make purchase decisions more easily.

When people purchase a product from an online store, they are usually provided product-related information such as product description, product images, and user reviews. Such information contains details of a product or other consumers' opinions, and it can help the consumers make purchase decisions. Often, product specifications are also provided to specify its features in an organized way, especially for high-technology products that consist of several electronic components. However, it is difficult to understand what the contents of product specifications imply when the consumers are unfamiliar with them. For example, when novice consumers read a digital camera's specifications, they probably do not have any idea what the value "TTL phase detection" of the feature "Auto Focus" means since they are not familiar with the feature value. Not only such consumers are unfamiliar with what the feature value is, but also they do not know how it truly means to them. In order to choose a product with the "right" value of a feature, they would like to hear direct experience from other consumers who own a product equipped with it, which may answer questions such as "is the feature value preferred by others?" and "is the feature considered important by others?"

[1]Part of this work has been published in [79].

Finding out what people have said about a feature or a feature value across different products is a laborious task. The problem is worsened by the fact that many new high-tech products increasingly have more new features. For example, a digital camera *Canon EOS 70D* has 79 features in CNET's product specifications page[2]. Thus, it is our goal to automatically augment product specifications through mining knowledge from product reviews and specifications across different products. Specifically, an augmented specification would show relevant review sentences, feature importance, and product-specific word list for each feature. An example of augmented specifications is shown in Figure 5.1.

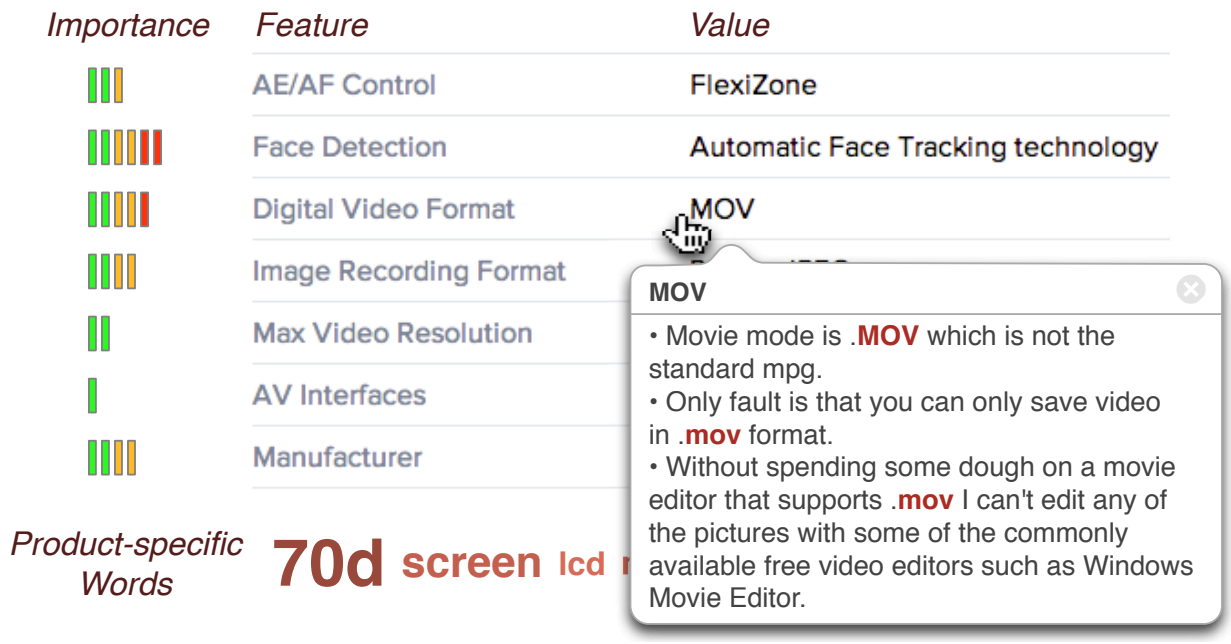


Figure 5.1: An example of augmented specifications for a digital camera *Canon EOS 70D*.

The augmented specifications can be very useful for consumers. For example, the following sentence is retrieved from a review of a product with the feature-value pair ("Battery – Supported Battery", "Canon LP-E6 Li-ion rechargeable battery"). *The 60D uses the LP-E6 battery like the 7D, which is a nice feature as this battery can often last through a full day of shooting.* Through reading such user experience, consumers can learn about the feature value, and it will help them choose a proper product without reading all the reviews of products with that battery. In addition, the augmented specifications can provide how important the feature "Battery – Supported Battery" is to other customers and provide what characteristics are special for a certain product.

To the best of our knowledge, no previous work has addressed the novel problem of mining relevant opinions for a feature value in specifications across different products. Despite widespread existence of

[2]http://www.cnet.com/products/canon-eos-70d/specs

product specifications on the Web, only a few researchers studied specifications [117, 7, 112, 103, 82], and the goals of them are different from ours. This work makes the following contributions:

1. We introduce and study a novel problem of mining product reviews (unstructured text) and product specifications (structured text) jointly to discover relevant review sentences to each feature value in product specifications.

2. We propose a new probabilistic topic model, SpecLDA, to solve the proposed mining problem. SpecLDA can also infer each feature's importance and each product's specific theme. SpecLDA is general so that it can be applied to mine other kinds of text data and the companion structured data for alignment of values in structured data with sentences in text data.

3. We create a new data set for evaluating the new task and conduct experiments to show that the proposed SpecLDA outperforms a related state-of-the-art model for extracting relevant review sentences.

The potential impact of our model is significant. The augmented specifications can benefit product manufacturers as well as consumers. After learning what customers say about a feature or a feature value of products in a current market, they may focus on important features and develop components desired by consumers. In addition, since our suggested approach can be applied to any data set consisting of unstructured text and specifications (key-value pairs) of entities, it can be employed in other problems.

5.2 Related Work

Finding opinions from a text data set have been widely studied, and several surveys summarized existing work [45, 59, 76]. Most of the studies performed research on product review [18, 37] or Weblog [68] data sets since people leave rich opinions on them. In order to find the object of opinions and to mine opinions in a more effective way, aspect-based opinion mining and summarization [37, 85, 101] has been studied as a main stream in the field. Our work is related to the aspect-based opinion mining as we retrieve a user's review text on a particular product aspect. To find the aspects of a product, many studies [115, 68, 96] applied topic models [8, 32], which find latent topics from a text corpus. While most of the existing topic model-based approaches find latent topics without constraints on the topics, we utilize prior knowledge on the topics so that the topics and the specifications can match. Lu and Zhai [63] also used semi-supervised topic modeling with pre-defined topics, but their goal is to find opinions for a product aspect (feature), while we find opinions for each value of a product feature so that we can mine knowledge about feature values across different products. Most existing studies in this line of research mine opinions on a product feature,

either pre-defined or latent, but we mine opinions for a smaller grain of topic, a feature value.

Although product specifications have been available in many e-commerce Web sites, only a limited number of studies employed them for product review analysis. Zhou and Chaovalit [117] established Ontology-Supported Polarity Mining (OSPM), which takes advantage of domain ontology database from IMDb[3], and their goal is sentiment classification on reviews. However, they studied only movie properties (features), not feature values. Bhattattacharya *et al.* [7] also used IMDb's structured data, but their goal is document categorization. Yu et al. [112] employed product specifications and reviews to build an aspect hierarchy, but they did not study feature values. Wang *et al.* [103] and Peñalver-Martínez *et al.* [82] also used product specifications to summarize product features, but neither of them studied feature values.

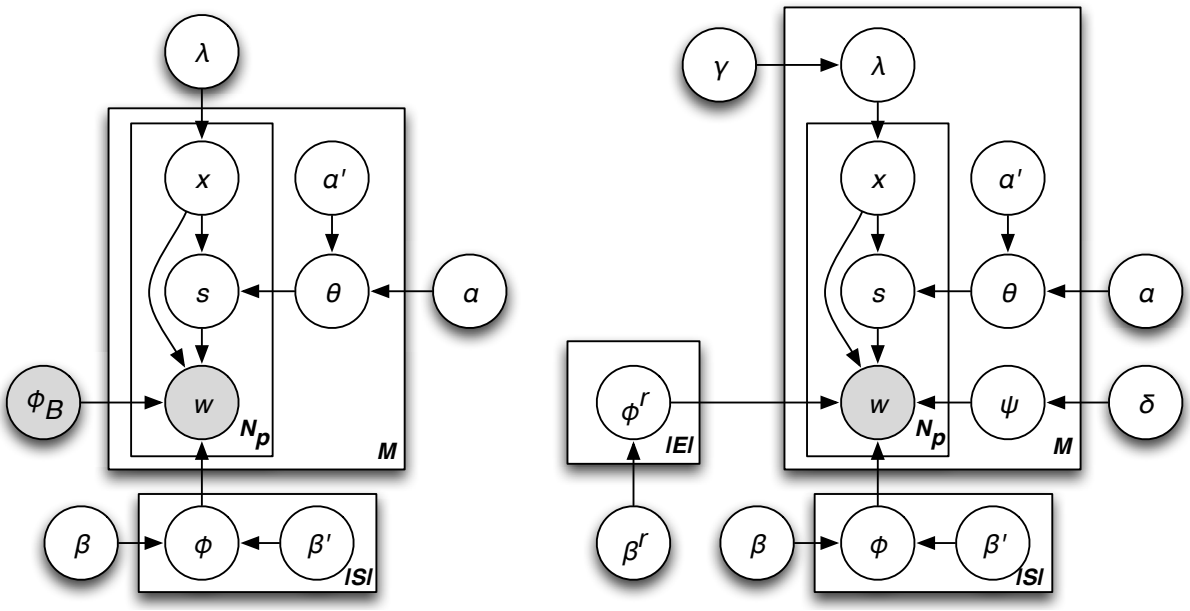
Modeling product reviews and specification simultaneously has been attempted in Duan *et al.* [22] with an extended PLSA model. However, the goal in Duan *et al.* [22] is to bridge the vocabulary gap in product search whereas our goal is to mine relevant review sentences to augment product specifications. Moreover, our proposed SpecLDA can better capture the feature structure in product specifications than the model used in Duan *et al.* [22] since their model does not consider hierarchy structure in specifications.

5.3 Problem Definition

The proposed new text mining problem is defined as follows. We are given M products $\boldsymbol{P} = \{P_1, ..., P_M\}$ with reviews $\boldsymbol{R}$, review sentences $\boldsymbol{T}$, and specifications $\boldsymbol{S}$. For each product p, there are specifications $\boldsymbol{S}_p$ and reviews $\boldsymbol{R}_p$ consisting of sentences $\boldsymbol{T}_p$. Specifications $\boldsymbol{S}_p$ of a product p is defined as $\boldsymbol{S}_p = \{s | s \in \boldsymbol{S}$ *and* s is part of $p\}$, where a specification s is a feature-value pair (f, u), and $\boldsymbol{S}$ is a set of all possible feature-value pairs. Our goal is (1) to mine a set of sentences $\{t_1, ..., t_k\}$ for each specification (f, u), (2) predict importance for each feature f, and (3) mine a set of product-specific words for each product p. Please refer to Figure 5.1 as an example of output that we hope to generate.

This is a new problem that has not been addressed yet in previous work, and it can be regarded as a step toward an interesting new kind of mining problems involving both text data and the associated structured data. The problem is challenging for the following reason. The vocabulary used in specifications and reviews for a feature or a feature value may be different. Even if the vocabulary used in the reviews are the same as that in specifications, it is not known which words of the vocabulary people prefer to use to indicate a certain feature value. If we just use words of the given feature value, we may miss many of the relevant review sentences. Moreover, the data space is sparse since there may be many feature values for each of many features. Thus, reliable estimation of specifications model is desired.

[3]http://www.imdb.com



(a) DuanLDA (b) SpecLDA–

Figure 5.2: Graphical representation of DuanLDA and SpecLDA–.

5.4 Methods

Generating augmented specifications using reviews is a new problem. There is no existing method that can be directly used to solve this problem. We propose to solve the problem by using a topic model for capturing specifications words in review text while imposing a prior defined based on the product specifications. This idea is similar to the extended PLSA model proposed in Duan *et al.* [22] for product search. We thus first discuss how we can adapt this model to solve our problem.

5.4.1 DuanLDA

In order to retrieve query-relevant products, Duan *et al.* [22] developed a probabilistic method that models product reviews and specifications. Their model can be regarded as a semi-supervised PLSA model with specifications as pre-defined topics and concatenated reviews as documents, which maximizes the following log-likelihood function of the whole data set:

$$l = \sum_{p \in \boldsymbol{P}} \sum_{w \in \boldsymbol{V}} c(w, r_p) \log \left[\lambda p(w|\boldsymbol{\phi}_B) + (1-\lambda) \sum_{s \in S} p(w|s) p(s|p) \right] \tag{5.1}$$

where $c(w, r_p)$ is a word count in concatenated reviews r_p for p, and $\boldsymbol{\phi}_B$ is a background language model. λ is a parameter for choosing either $\boldsymbol{\phi}_B$ or the predefined topics $\boldsymbol{S}_p$, and $\boldsymbol{V}$ is a vocabulary set for the corpus.

As discussed in [8, 113], LDA has several advantages over PLSA, so we convert their model to LDA version and call it DuanLDA. The graphical representation of DuanLDA is shown in Figure 5.2a. There are M product documents, where each document is a concatenated review for p, and there are N_p words for each document. s is a specification (a feature-value pair) topic, and there are $|\boldsymbol{S}|$ possible specifications. The generative story for each word is as following. When an author writes a review word $w_{p,i}$ at ith position of product document p, the author chooses a background topic or specification topics according to switch $x_{p,i}$, which is determined by a parameter λ. If the background topic is chosen, $w_{p,i}$ is drawn from background language model $\boldsymbol{\phi}_B$; otherwise, a specification $s_{p,i}$ is chosen according to $\boldsymbol{\theta}_p$, and $w_{p,i}$ is chosen according to $\boldsymbol{\phi}_{s_{p,i}}$. To incorporate pre-defined topics into LDA, product-specific topic distributions $\boldsymbol{\alpha}'_p$ and topic-specific word distributions $\boldsymbol{\beta}'_z$ are used to draw $\boldsymbol{\theta}_p$ and $\boldsymbol{\phi}_{s_{p,i}}$. Specifically, $\boldsymbol{\theta}_p$ is drawn from Dirichlet($\alpha\boldsymbol{\alpha}'_p$) and $\boldsymbol{\phi}_{s_{p,i}}$ is drawn from Dirichlet($\beta\boldsymbol{\beta}'_{s_{p,i}}$), and how to generate those prior distributions is explained later in this section.

The document language model for DuanLDA is defined by

$$p_{lda}(w|p,\lambda,\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\phi}}) = \lambda p(w|\boldsymbol{\phi}_B) + (1-\lambda)\sum_{s=1}^{|\boldsymbol{S}|} p(w|s,\hat{\boldsymbol{\phi}})p(s|p,\hat{\boldsymbol{\theta}}) \tag{5.2}$$

The probability of $x_{p,i}=0$, which chooses background language model, is determined by λ and the corpus $\boldsymbol{W}$, which is

$$\begin{aligned} p(x_{p,i}=0|\boldsymbol{W},\lambda) &\propto \lambda p(w_{p,i}|\boldsymbol{\phi}_B) \\ &\propto \lambda \frac{N_{w_{p,i}}}{\sum_{w'\in\boldsymbol{V}} N_{w'}} \end{aligned} \tag{5.3}$$

where N with superscript and/or subscript means the number of words satisfying the superscript and subscript conditions. A Collapsed Gibbs sampling formula to choose $s_{p,i}$ when $x_{p,i}=1$ is defined as

$$\begin{aligned} &p(x_{p,i}=1, s_{p,i}=z|w_{p,i},\boldsymbol{W}_{\setminus p,i},\boldsymbol{S}_{\setminus p,i},\lambda,\alpha,\boldsymbol{\alpha}'_p,\beta,\boldsymbol{\beta}'_{s_{p,i}}) \\ &\quad\propto (1-\lambda)p(w_{p,i}|s_{p,i}=z,\boldsymbol{W}_{\setminus p,i},\boldsymbol{S}_{\setminus p,i},\beta,\boldsymbol{\beta}'_z)p(s_{p,i}=z|\boldsymbol{S}_{\setminus p,i},\alpha,\boldsymbol{\alpha}'_p) \\ &\quad\propto (1-\lambda)\frac{N^{\setminus p,i}_{w_{p,i}|z}+|\boldsymbol{V}|\beta\beta'_{z,w_{p,i}}}{N^{\setminus p,i}_{z}+|\boldsymbol{V}|\beta}\frac{N^{\setminus p,i}_{z|p}+K\alpha\alpha'_{p,z}}{N^{\setminus p,i}_{x=1|p}+K\alpha} \end{aligned} \tag{5.4}$$

where $\setminus p,i$ exclude a word at ith position of p.

Prior Generation In order to automatically generate priors $\boldsymbol{\beta}'_z$ for each topic z, we take an approach similar to that in [22]. However, the resulting distribution from their prior generation is quite even, so it

does not distinguish important words from unimportant words well. Thus, we assume the prior words follow Zipf's law distribution and adjust $p(w|f)$ according to it. Specifically, from the prior $p(w|f)$ obtained in [22], we define new word prior $p'(w|f)$ as

$$p'(w|f) = \begin{cases} \frac{p(w|f)}{\sum_{w \in \boldsymbol{V}(f)} p(w|f)} \sum_{i=1}^{|\boldsymbol{V}(f) \cap \boldsymbol{V}|} Zipf(i) & \text{if } w \in \boldsymbol{V}(f) \\ Zipf(rank_f(w) + |\boldsymbol{V}(f) \cap \boldsymbol{V}|) & \text{otherwise} \end{cases} \tag{5.5}$$

where $\boldsymbol{V}(f)$ is a vocabulary in f, $\boldsymbol{V}$ is a vocabulary in the review corpus, $rank_f(w)$ is w's rank in $p(w|f)$ excluding words in $\boldsymbol{V}(f)$, and Zipf's law distribution function $Zipf(i)$ is defined as $Zipf(i) = \frac{1/i^s}{\sum_{n=1}^{|\boldsymbol{V}|} 1/n^s}$, where s is a parameter characterizing the distribution. Basically, $p'(w|f)$ keeps the rankings in $p(w|f)$ but substitutes probabilities of non-feature words with Zipf's probability. $p(w|f)$ of feature words are redistributed to $p'(w|f)$ having sum of them equal to Zipf's probability sum for first $|\boldsymbol{V}(f) \cap \boldsymbol{V}|$ words. Then, we assign $\beta'_{s,w} = p'(w|f)$ for s whose feature is f. Also, we generate document-topic prior $\boldsymbol{\alpha}'$ based on specifications in each product; if a feature-value pair s is not present in a product p, we assign zero to $\alpha'_{p,s}$, and otherwise, we assign $\alpha'_{p,s}$ a probability, which is uniform among all present feature-value pairs.

5.4.2 SpecLDA: A topic model for joint mining of product reviews and specifications

DuanLDA has several deficiencies: (1) it considers only specification topics, (2) the prior amount ($|\boldsymbol{V}|\beta$) is uniform for all topics, (3) it does not fully take advantage of the specifications structure. We develop SpecLDA– (Figure 5.2b) that improves the points (1) and (2) and mines product-specific words. Then, we we further address point (3) and propose SpecLDA.

SpecLDA–

Review Topics Product reviews may have topics that are not in specifications; for example, value, design, or ease of use is not listed in specifications, but they may be mentioned in reviews. DuanLDA expects them to be captured by background language model, but it assumes that every product document has the same proportion of background topic, which may not be true. We thus remove background topic and add $|\boldsymbol{E}|$ review topics, resulting in all topics $\{s_1, \ldots, s_{|\boldsymbol{S}|}, s_{|\boldsymbol{S}|+1}, \ldots, s_{|\boldsymbol{S}|+|\boldsymbol{E}|}\}$, similar to those in [63]. Now, $\boldsymbol{\alpha}'_p$ contains uniform prior among present specifications and review topics, and zero for absent specifications. If the drawn topic $s_{p,i}$ belongs to specifications, it works the same as DuanLDA does. On the other hand, if $s_{p,i}$ belongs to review topics ($\boldsymbol{E}$), the word $w_{p,i}$ is drawn from $\boldsymbol{\phi}^r_{s_{p,i}}$, which is drawn from Dirichlet distribution with a symmetric prior β^r.

Prior Regularization Each specification topic s has its estimated topic size N_s. If the topic size N_s is relatively too small or too big compared to amount of prior, $|\boldsymbol{V}|\beta$, the topic s will rely too much or too little on the prior $\boldsymbol{\beta}'_s$. If a topic relies too much on prior, then the topic will just follow the word distribution of $\boldsymbol{\beta}'_s$, and if a topic relies too little on prior, it is likely to bear other themes that are unrelated to prior so that the topic is corrupted. Therefore, we need to regularize the prior size according to topic sizes, which is done similarly as in [95, 63]. We initialize prior size coefficient β with a big number, and we introduce prior size controllers $\{\eta_1, \ldots, \eta_{|\boldsymbol{S}|}\}$, each of which repeatedly decays by decay factor ζ if the topic size is too little.

Please note that we do not explicitly insert regularization variables to the Gibbs sampling formulas for SpecLDA for simplicity.

Product-specific Topics We add a product-specific topic $\boldsymbol{\psi}_p$ for each product p in order to capture how p is different from other products in reviews. In other words, if a word is closer to a product-specific topic than to any other topics, it is likely to be assigned with the product-specific topic. When a review author writes a word $w_{p,i}$ for a product p, the author first chooses between product-specific topic and specification topics according to $\boldsymbol{\lambda}_p$, which is drawn from Beta distribution with a symmetric prior γ. If the product-specific topic is chosen ($x_{p,i} = 0$), $w_{p,i}$ is drawn from $\boldsymbol{\psi}_p$, which is drawn from Dirichlet distribution with a symmetric δ.

SpecLDA

Feature-value pairs with the same feature share the feature information, but DuanLDA models them individually. We suggest to form a hierarchy among specifications, where we separate feature topics and value topics and share the feature topics for feature-value pairs with the same feature. By forming such hierarchy, we can expect SpecLDA to have more reliable topic estimation since it shares the feature topics and connects feature-value pairs who belong to the same feature. In addition, we can capture a user's preference on choosing feature-related or value-related words to indicate a feature-value pair. Another deficiency in Duan *et al.*'s model is that it imports priors only from feature words, not from value words. However, we believe that value word priors are also important, so we employ them as well as feature word priors. Value word priors are generated in the same way as feature word priors are.

The graphical representation of SpecLDA is depicted in Figure 5.3. For each feature f of all possible features $\boldsymbol{F}$, there are possible values $\boldsymbol{U}^f$. To separate a feature from feature values, a feature variable f is separated from the value variable u^f, which is a possible value for f. Also, the feature value topics $\boldsymbol{\omega}$ is introduced to separate them from feature topics $\boldsymbol{\phi}$. The generative story is as following. Choosing a word for product-specific is the same as in SpecLDA–. If a product feature is chosen by $x_{p,i}$, the author chooses

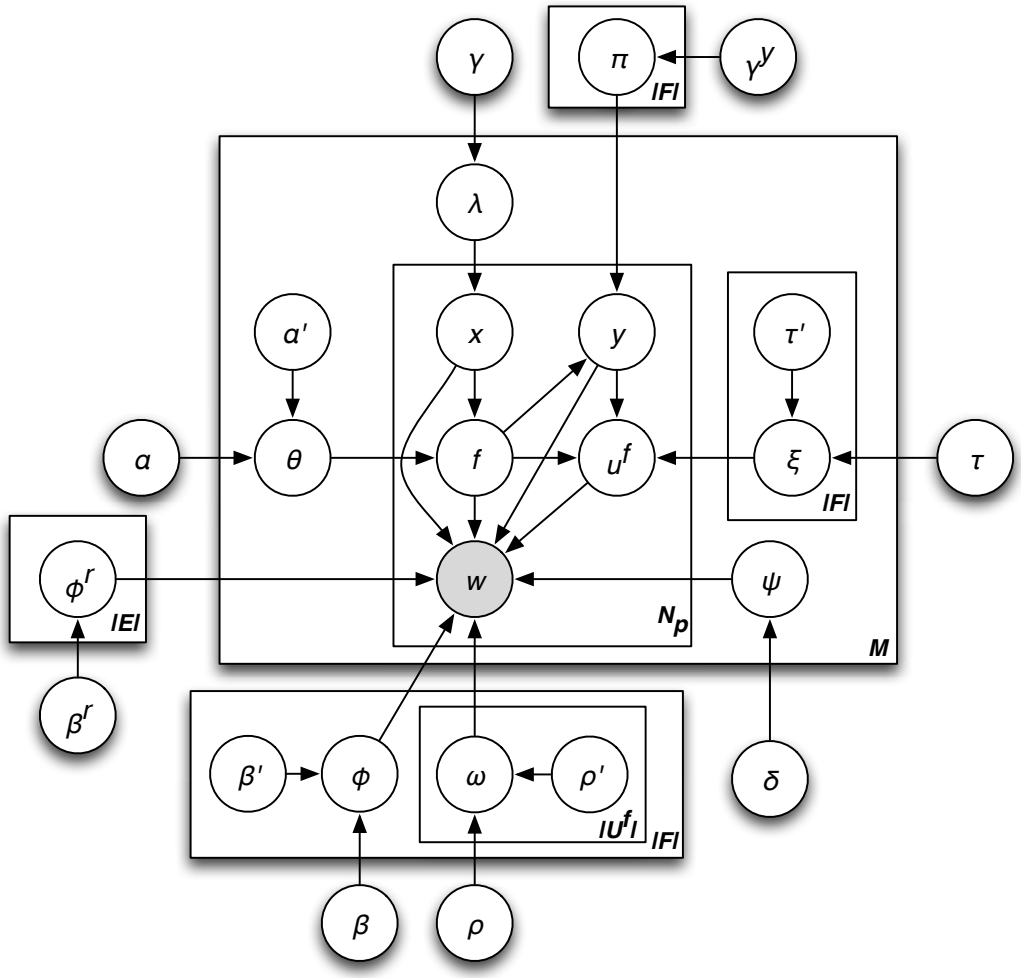


Figure 5.3: SpecLDA

a feature $f_{p,i}$ from the possible feature set $\{f_1, \ldots, f_{|\boldsymbol{F}|}, f_{|\boldsymbol{F}|+1}, \ldots, f_{|\boldsymbol{F}|+|\boldsymbol{E}|}\}$, which is a set of feature and review topics, according to $\boldsymbol{\theta}_p$, which is drawn from Dirichlet distribution with $\alpha\boldsymbol{\alpha}'_p$. If $f_{p,i}$ belongs to review features, $w_{p,i}$ is drawn from multinomial distribution $\boldsymbol{\phi}^r_{f_{p,i}}$, which is drawn from Dirichlet distribution with a symmetric prior β^r. If the chosen feature $f_{p,i}$ belongs to specifications features, the author again chooses to write a feature or a value word using switch $y_{p,i}$ according to $\boldsymbol{\pi}_{f_{p,i}}$, which is drawn from beta distribution with a symmetric prior γ^y. If the author chooses to write a feature word, $w_{p,i}$ is chosen according to $\boldsymbol{\phi}_{f_{p,i}}$, which is drawn from Dirichlet distribution with $\beta\boldsymbol{\beta}'_{f_{p,i}}$. Otherwise, the author further chooses value $u^f_{p,i}$ for $f_{p,i}$ according to $\boldsymbol{\xi}_{p,f_{p,i}}$, which is drawn from Dirichlet distribution with $\tau\boldsymbol{\tau}_{p,f_{p,i}}$. With the chosen feature value $u^f_{p,i}$, the author chooses a word according to $\boldsymbol{\omega}_{f_{p,i},u^f_{p,i}}$, which is drawn from Dirichlet distribution with $\rho\boldsymbol{\rho}'_{f_{p,i},u^f_{p,i}}$. This process is repeated for all review words of all products.

The document language model of SpecLDA is thus:

$$
\begin{aligned}
&p_{lda}(w|p,\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\xi}},\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\phi}}^{r},\hat{\boldsymbol{\omega}},\hat{\boldsymbol{\psi}}) \\
&\quad = p(x=0|\hat{\boldsymbol{\lambda}}_p)p(w|\hat{\boldsymbol{\psi}}_p) + p(x=1|\hat{\boldsymbol{\lambda}}_p)\big[\sum_{f\in\boldsymbol{E}} p(w|\hat{\boldsymbol{\phi}}_f^{r})p(f|\hat{\boldsymbol{\theta}}_p) + \sum_{f\in\boldsymbol{F}_p} p(f|\hat{\boldsymbol{\theta}}_p)p(w|f,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\xi}},\hat{\boldsymbol{\omega}})\big]
\end{aligned} \tag{5.6}
$$

where

$$
p(w|f,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\xi}},\hat{\boldsymbol{\omega}}) = p(y=0|\hat{\boldsymbol{\pi}}_f)p(w|\hat{\boldsymbol{\phi}}_f) + p(y=1|\hat{\boldsymbol{\pi}}_f)\sum_{u\in\boldsymbol{U}^f} p(u|\hat{\boldsymbol{\xi}}_f)p(w|\hat{\boldsymbol{\omega}}_{f,u}) \tag{5.7}
$$

and the Gibbs sampling formula for learning when product-specific topic is used ($x=0$) is

$$
\begin{aligned}
p(x_{p,i}=0|w_{p,i},\boldsymbol{W}_{\setminus p,i},\boldsymbol{X}_{\setminus p,i},\gamma,\delta) &\propto p(x_{p,i}=0|\boldsymbol{X}_{\setminus p,i},\gamma)p(w_{p,i}|\boldsymbol{X}_{\setminus p,i},\boldsymbol{W}_{\setminus p,i},\delta) \\
&\propto \frac{N^{\setminus p,i}_{x=0|p}+\gamma}{N_p-1+2\gamma}\frac{N^{\setminus p,i}_{w_{p,i}|x=0}+\delta}{N^{\setminus p,i}_{x=0}+|\boldsymbol{V}|\delta}
\end{aligned} \tag{5.8}
$$

To learn when we choose a review topic or a feature topic f, the formula is defined as

$$
\begin{aligned}
&p(x_{p,i}=1,f_{p,i}=z,y_{p,i}=0|w_{p,i},\boldsymbol{W}_{\setminus p,i},\boldsymbol{X}_{\setminus p,i},\boldsymbol{F}_{\setminus p,i},\boldsymbol{E}_{\setminus p,i},\boldsymbol{Y}_{\setminus p,i},\Omega) \\
&\quad\propto p(x_{p,i}=1|\boldsymbol{X}_{\setminus p,i},\Omega) \\
&\qquad p(f_{p,i}=z|\boldsymbol{F}_{\setminus p,i},\boldsymbol{E}_{\setminus p,i},\Omega) \\
&\qquad p(y_{p,i}=0|z,\boldsymbol{Y}_{\setminus p,i},\boldsymbol{F}_{\setminus p,i},\boldsymbol{E}_{\setminus p,i},\Omega) \\
&\qquad p(w_{p,i}|z,\boldsymbol{W}_{\setminus p,i}\boldsymbol{F}_{\setminus p,i},\boldsymbol{E}_{\setminus p,i},\boldsymbol{Y}_{\setminus p,i},\Omega) \\
&\quad\propto \begin{cases}
\frac{N^{\setminus p,i}_{x=1|p}+\gamma}{N_p-1+2\gamma}\times\frac{N^{\setminus p,i}_{z|p}+|\boldsymbol{F}|\alpha\alpha'_{p,z}}{N^{\setminus p,i}_{x=1|p}+K\alpha}\times\frac{N^{\setminus p,i}_{y=0|z}+\gamma^{y}}{N^{\setminus p,i}_{z}+2\gamma^{y}}\times\frac{N^{\setminus p,i}_{w_{p,i}|z,y=0}+|\boldsymbol{V}|\beta\beta'_{z,w_{p,i}}}{N^{\setminus p,i}_{z,y=0}+|\boldsymbol{V}|\beta} & \text{if } z\in\boldsymbol{F} \\
\frac{N^{\setminus p,i}_{x=1|p}+\gamma}{N_p-1+2\gamma}\times\frac{N^{\setminus p,i}_{z|p}+\alpha}{N^{\setminus p,i}_{x=1|p}+K\alpha}\times 1\times\frac{N^{\setminus p,i}_{w_{p,i}|z,y=0}+\beta^{r}}{N^{\setminus p,i}_{z,y=0}+|\boldsymbol{V}|\beta^{r}} & \text{if } z\in\boldsymbol{E}
\end{cases}
\end{aligned} \tag{5.9}
$$

where Ω is all priors and K is the number of all topics ($|\boldsymbol{S}|+|\boldsymbol{E}|$). The probability that a feature and a feature value are chosen ($f_{p,i}=z$, $u_{p,i}=j$) is defined as

$$
\begin{aligned}
&p(x_{p,i}=1, f_{p,i}=z,\ y=1, u_{p,i}=j | w_{p,i}, \boldsymbol{W}_{\setminus p,i}, \boldsymbol{X}_{\setminus p,i}, \boldsymbol{F}_{\setminus p,i}, \boldsymbol{Y}_{\setminus p,i}, \boldsymbol{U}_{\setminus p,i}, \Omega) \\
&\quad \propto p(x_{p,i}=1 | \boldsymbol{X}_{\setminus p,i}, \Omega) \\
&\qquad p(f_{p,i}=z | \boldsymbol{X}_{\setminus p,i} \boldsymbol{F}_{\setminus p,i}, \boldsymbol{E}_{\setminus p,i}, \Omega) \\
&\qquad p(y_{p,i}=1 | z, \boldsymbol{Y}_{\setminus p,i}, \boldsymbol{F}_{\setminus p,i}, \boldsymbol{E}_{\setminus p,i}, \Omega) \\
&\qquad p(u_{p,i}=j | z, \boldsymbol{Y}_{\setminus p,i}, \boldsymbol{F}_{\setminus p,i}, \boldsymbol{U}_{\setminus p,i}, \Omega) p(w_{p,i} | z, j, \boldsymbol{W}_{\setminus p,i}, \boldsymbol{U}_{\setminus p,i}, \Omega) \\
&\quad \propto \begin{cases} \frac{N^{\setminus p,i}_{x=1|p}+\gamma}{N_p-1+2\gamma} \times \frac{N^{\setminus p,i}_{z|p}+|\boldsymbol{F}|\alpha\alpha'_{p,z}}{N^{\setminus p,i}_{x=1|p}+K\alpha} \times \frac{N^{\setminus p,i}_{y=1|z}+\gamma^y}{N^{\setminus p,i}_{z}+2\gamma^y} \times \frac{N^{\setminus p,i}_{j|p}+|\boldsymbol{U}^f|\tau\tau'_{p,z,j}}{N^{\setminus p,i}_{z,y=1|p}+|\boldsymbol{U}^f|\tau} \times \frac{N^{\setminus p,i}_{w_{p,i}|j}+\boldsymbol{V}\rho\rho'_{z,j,w_{p,i}}}{N^{\setminus p,i}_{j}+\boldsymbol{V}\rho} & \text{if } z \in \boldsymbol{F} \\ 0 & \text{if } z \in \boldsymbol{E} \end{cases}
\end{aligned}
\tag{5.10}
$$

where $|\boldsymbol{U}^f|$ is the number of all possible feature values for the feature f. Regularization is applied to priors for both feature words and feature value words.

Feature Importance The importance of a feature may be useful for a novice customer who wants to know which features are considered important according to reviews. We assume feature importance is determined by how often the features are mentioned in reviews. In SpecLDA, the assignments of feature topics to words can be counted and used as feature popularity. A feature popularity of a feature f is defined as $popularity(f) \propto \frac{N_f}{N_{x=1}}$ where N_f is the number of words assigned with f, and $N_{x=1}$ means the number of words assigned with any features.

5.5 Experiments

In this section, we describe how we perform experiments in detail. In particular, we describe how we collect the data set including human judgments, how we set parameters, and how we retrieve opinions with the proposed topic model. Then, we analyze the experiment results qualitatively and quantitatively.

5.5.1 Data Set

It is required for our task to obtain reviews and specifications for products, and these kinds of data are available in several web sites such as Amazon.com, BestBuy.com, and CNET.com. Among them, we chose CNET.com because they have a reasonable amount of user reviews and relatively well-organized specifications. We crawled the reviews and specifications that were available on February 22, 2012 in digital camera category. We chose the digital camera category since a digital camera is a high-technology product with a

lot of features, but our models can also be applied to a product category without many features as well. To pre-process the review text, we performed sentence segmentation, word tokenization, and lemmatization using Stanford CoreNLP [66] version 1.3.5. We lowered word tokens and removed punctuation. Then, we removed stopwords provided by Mallet [67], and we omitted word tokens that appear in less than five reviews or more than 30% of the reviews since they are barely informative. We also pre-processed specifications data. We removed feature values that appear in less than five products. Then, we split each feature and feature value text into word tokens, and we lowered the word tokens.

The resulting data contain 1,153 products that have specifications and at least one review. The total number of reviews is 11,870, with the total number of sentences being 88,527, where each sentence has 7.74 word tokens on average. The reviews in the same product are concatenated to form a single product document, yielding 1,153 documents with average length being 594.15. In the specifications data, we have 2,320 distinct feature-value pairs S in total, including 124 distinct features $\boldsymbol{F}$.

Table 5.1: Parameter setting for topic models.

Parameter	DuanLDA	SpecLDA–	SpecLDA
λ	0.3	-	-
α	$\frac{50}{\|S\|}$	$\frac{50}{\|S\|+\|E\|}$	$\frac{50}{\|F\|+\|E\|}$
β	0.1	10.0	10.0
β^r	-	0.01	0.01
γ	-	0.5	0.5
δ	-	0.0001	0.0001
ζ	-	0.9	0.9
ϵ_{pp}	-	0.5	0.5
ϵ_{ts}	-	50	50
τ	-	-	0.01
ρ	-	-	10.0
γ^y	-	-	50

Parameter Setting For all the suggested LDA models, we use five Markov chains with 2,000 Gibbs sampling iterations each, where each chain is randomly initialized. Parameter values are empirically set and shown in Table 5.1, where "-" means not available for the model, and these values are used for all experiments unless specified otherwise.

As shown, SpecLDA– and SpecLDA use more parameters than DuanLDA. However, many of the additional parameters can be easily set. For example, we can just give a small number to γ, which means very weak supervision on the Bernoulli distribution. The decay factor ζ can be set to a value close to 1.0. We can assign high enough values for initial β and ρ. For DuanLDA, we set β that gives the highest evaluation scores while the topics hold the original specifications well; we measure topic corruption rate based

on KL-Divergence between the original specification words and the estimated topics and try to maintain very similar level of corruption rate for the suggested methods. β^r and δ should be set depending on the sizes of review topics and product-specific topics, respectively, and τ can be set depending on the size of feature-value topics. ϵ_{pp} and ϵ_{ts} can be set depending on how we want to control prior knowledge; if we want the topics more like prior words, we can set ϵ_{pp} close to 1.0 and ϵ_{ts} very high. γ^y is a smoothing parameter on choosing feature or value topics, so greater γ^y results in more even preference.

5.5.2 Mining Opinions by Sentence Retrieval

Once we learn the topic models, we use the estimated document language models, $p_{lda}(w|p)$, to mine opinion sentences for a specification s. To retrieve text using a document language model, we exploit query likelihood retrieval model [6], one of the standard ad-hoc retrieval method. Fortunately, in our problem setting, we can take advantage of specifications to filter out some of unrelated sentences; if a sentence t^p is from a product p's reviews, and s is not in p's specifications S_p, we can ignore t^p. The relevance score of t^p for s is thus defined as

$$score(s, t^p) = \begin{cases} 0 \quad \text{, if } s \notin \boldsymbol{S}_p \\ \prod_{w \in s} \left[(1-\chi) p_{lda}(w|t^p) + \chi p(w|\boldsymbol{W}) \right], \text{ otherwise} \end{cases} \tag{5.11}$$

where $p(w|\boldsymbol{W})$ is a background language model that is estimated by $\frac{count(w)}{|\boldsymbol{W}|}$, where $count(w)$ is the count of w in corpus $\boldsymbol{W}$. χ is a parameter to give a non-zero value to $p_{lda}(w|t^p)$, which is a standard smoothing technique.

Our goal is to mine opinion "sentences", while the suggested topic models estimate "document" language models. We can extend LDA to model each sentence, but it will require too many variables since the number of sentences is usually way greater than the number of documents. Thus, we convert estimations from document-level to sentence-level. Language model $p(w|t^p)$ for a sentence t in a product document p is thus defined as:

$$p_{lda}(w|t^p) = \sum_{z=1}^{K} \frac{\hat{N}_{w|z} + |\boldsymbol{V}|\beta\beta'_{z,w}}{\hat{N}_z + |\boldsymbol{V}|\beta} \frac{\hat{N}_{z|t} + \frac{|t|}{|p|} K\alpha\alpha'_{p,z}}{\hat{N}_t + \frac{|t|}{|p|} K\alpha} \tag{5.12}$$

where the size of topic-document prior $K\alpha$ is re-sized by the proportion of length of t to that of p since topic-document prior depends on the size of document. The same conversion technique is used for other priors that depends on document length in this work.

Query Expansion by Topic Models In addition to mining opinions based on original specification words, we also try mining opinions with query expansion. One of the advantages of using topic models

when topics represent queries is that the estimated topics can be used to expand the original queries. To use a topic as a specification query, we employ KL-divergence model, which is a generalization of the query likelihood model, and its derivation is well explained in [113]. Following the derivation, instead of $c(w, s)$ in formula (5.11), we put $p'(w|s)$, which is a query language model. The query language model we use is the interpolation of the original query model and the estimated topic model, and it is defined as

$$p'(w|s) = (1 - \mu)p(w|s) + \mu p(w|\hat{\phi}_s) \tag{5.13}$$

where $p(w|s) = \frac{c(w,s)}{|s|}$, and $p(w|\hat{\phi}_s)$ is the estimated topic language model for the specification s.

5.5.3 Qualitative Analysis

Table 5.2: Top 20 words of a topic for a specification ("Display – Type", "2.5 in. LCD Display").

DuanLDA	SpecLDA–	SpecLDA–_V	SpecLDA
display	digital	lcd	lcd
type	lcd	display	2.5
phtography	2.5	2.5	display
font	display	screen	screen
florescent	huge	type	large
informative	technology	large	inch
channel	expensive	inch	monitor
triple	type	monitor	articulate
resultant	large	phtography	1.8
lcd	outstanding	font	230,000
printout	icon	crack	panel
a10	silver	symbol	bright
colorful	follow	1.8	icon
information	rubber	icon	rotatable
info	phtography	bright	sunlight
horizontal	salesman	brightness	brightness
picture/video	font	articulate	viewfinder
infinite	info	salesman	tilt
2aa	blessing	range	viewer
3.0	symbol	informative	flippable

Specification Topics The top words in the specification topic ("Display – Type", "2.5 in. LCD Display") for each model are listed in Table 5.2. SpecLDA–_V is a value word prior-added version of SpecLDA–. The word distribution is extracted from $\hat{\phi}$ of DuanLDA, SpecLDA–, and SpecLDA–_V, and $\pi_0\hat{\phi} + \pi_1\hat{\omega}$ of SpecLDA. As shown in the table, words in DuanLDA do not show reasonable relevance to the feature value "2.5 in. LCD Display" since it does not utilize value word priors and the prior amount is fixed. On the other hand, SpecLDA– attracted a few feature value-related words with dynamic prior amount. SpecLDA–_V, which uses value prior words, drew more value-related words, but it still keeps a few unrelated words. Most

of the twenty words SpecLDA attracted seem to be related to the feature value because the feature topic and value topics form a cluster, resulting in better topic estimation.

Table 5.3: Examples of product-specific topics.

Logitech ClickSmart 310	Canon PowerShot SX10 IS	Canon PowerShot S2 IS
webcam	lens	zoom
cam	zoom	video
video	sx10	s2
image	stamp	movie
digital	20x	cap
computer	video	canon
flash	cap	mode
web	slr	shot
cheap	date/time	sony
figure	superzoom	12x

Product-specific Topics In order to inform customers what is special about a product, we capture a product-specific topic ψ for each product, and the examples of $\hat{\boldsymbol{\psi}}$ are listed in Table 5.3. In the table, there are several words related to webcams for the product *Logitech ClickSmart 310*, which is actually a webcam that is a rare product category in the data set. *Canon PowerShot SX10 IS* features 20X optical zoom, which is indeed rare (in only five products) and relatively very high performance in the data set, and $\hat{\boldsymbol{\psi}}$ captures related words such as "lens", "zoom", "20x", and "superzoom" quite well. From the top words of *Canon PowerShot S2 IS*, we can see that zoom and video are special. The following sentences from CNET's editor's review, which is not included in our data set, support why those words are highlighted for the product.

> The S2's VGA *movie mode*, which now supports stereo audio, is quite good, with a top resolution of 640x480 at 30fps. Unlike many cameras with similar *movie-capture modes*, the *Canon* lets you use the *zoom*, which operates very quietly, and the IS while capturing *video*.

As shown in the examples, the product-specific topics capture special characteristics of products reasonably well. However, since each product-specific topic is estimated from reviews of a single product, not enough words are assigned with the topic if there are not many review texts for the product.

Feature Importance By our assumption that more important features are mentioned more frequently in reviews, we compute feature popularity scores and show the ten most popular features out of 128 possible features in Table 5.4. Surprisingly, "Additional Features – Additional Features" is ranked first in the list. The feature has the greatest number of distinct values (170) so that it is present in almost all products,

Table 5.4: Ten most popular features.

Most Popular Features
Additional Features – Additional Features
Exposure & White Balance – Shooting Program
Miscellaneous – Included Accessories
Exposure & White Balance – Light Sensitivity
Camera Flash & Flash Modes
Battery – Supported Battery
Memory / Storage – Supported Memory Cards
Lens System – Type
Lens System – Zoom Adjustment
Software

and the word "feature" is one of the most frequent words in the data set. Therefore, many words correlated to the word "feature" are assigned with this feature. Also, the feature "Lens System – Zoom Adjustment" actually contains words more related to zoom capability than zoom adjustment, due to the fact that zoom adjustment is barely mentioned in reviews so that words related to zoom capability are attracted to the topic. This is limitation of our model, and one should consider filtering out those very unpopular features. Other listed features are regarded reasonably important when people purchase a digital camera.

5.5.4 Quantitative Analysis

Human-labeled Data

In order to quantitatively evaluate how well the suggested methods mine opinion sentences for feature values, we need to make a gold standard data set labeled by humans. However, labeling 88,527 sentences by each of multiple annotators is too expensive. A domain expert was asked to choose the twelve most important features by looking at specifications and word counts in the whole corpus. Then, the words in the possible feature-value pairs were used to retrieve candidate sentences (13,671) by the models in Section 5.4. Each of the candidate sentences was labeled whether the sentence is relevant to a specification by three annotators at a crowdsourcing service CrowdFlower[4]. The agreement rate was 0.926, and Cohen's kappa coefficient was 0.678, where values between 0.61-0.80 are regarded as substantial agreement among annotators [50]. After omitting non-agreed sentences and queries with less than 20 true relevant sentences, we have 44 queries with 1,251 relevance data on average for each query.

[4]http://www.crowdflower.com/

Evaluation Metric

We use Mean Average Precision (MAP) as an evaluation metric, which is a mean of average precision for each query. The metric basically measures general retrieval performance for multiple queries, and if we have more relevant documents in high ranks, then the score becomes higher. Specifically, we use MAP@k that computes mean of average precision at the top k retrieved sentences for each query, and we set k as 5, 10, and 20 to see how well the models work in different user satisfaction levels.

5.5.5 Result Analysis

We evaluate the sentences retrieved for specification queries. For all the models, we use the original queries, which are concatenated strings of feature words and value words. For SpecLDA– and SpecLDA, we set the number of review topics $|E| = 5$. Table 5.5 shows evaluation results for the baseline model DuanLDA and our new models SpecLDA– and SpecLDA. † is used to mark models if the improvement is statistically (paired t-test with p=0.05) significant in all measures from the DuanLDA. SpecLDA–, which uses prior regularization, significantly improves DuanLDA on all measures. SpecLDA outperforms the baseline model significantly on all measures, and it also outperforms SpecLDA–, especially on MAP@20.

Table 5.5: MAP evaluation results for finding sentences relevant to a specification. Amount of improvement from DuanLDA is in parenthesis.

	MAP@5	MAP@10	MAP@20
DuanLDA	0.927	0.851	0.780
SpecLDA–†	0.970 (4.6%)	0.894 (5.1%)	0.812 (4.1%)
SpecLDA†	**0.986** (6.4%)	**0.917** (7.8%)	**0.849** (8.8%)

Results in Table 5.6 show evaluation results when the query language models are expanded by topics. DuanLDA_V is a version of DuanLDA that adds value word priors. Models in the upper part use only feature word priors, and those in the lower part use value word priors as well. Comparing results in Table 5.5 and those in Table 5.6, we can see that query expansion by topic models indeed helps us mine opinion sentences especially with SpecLDA. When value word priors are not used, SpecLDA– significantly outperforms DuanLDA. SpecLDA–_V significantly outperforms DuanLDA_V, and SpecLDA even improves SpecLDA–_V, which again means SpecLDA's hierarchy structure is effective.

5.6 Conclusion and Future Work

In this work, we studied the problem of automatically augmenting product specifications by jointly modeling product reviews and specifications. In specific, we defined the novel problem of relevant sentence retrieval

Table 5.6: Evaluation results by query expansion. Amount of improvement from DuanLDA (upper part) and DuanLDA_V (lower part) is in parenthesis. † is used to indicate statistical significance on all measures against DuanLDA (upper part) and DuanLDA_V (lower part).

	μ	MAP@5	MAP@10	MAP@20
DuanLDA	0	0.927	0.851	0.780
SpecLDA–†	0.2	**0.978** (6%)	**0.893** (5%)	**0.822** (5%)
DuanLDA_V	0.7	0.964	0.891	0.828
SpecLDA–_V†	1.0	**0.986** (2%)	0.952 (7%)	0.880 (6%)
SpecLDA†	0.7	**0.986** (2%)	**0.969** (9%)	**0.905** (9%)

for feature values and suggested a novel approach that is shown to effectively model reviews and specifications. We also demonstrated the inference of feature importance and product-specific words, which may be important for consumers. The potential impact of the augmented specifications is significant since both consumers and manufacturers may benefit from them.

There are limitations in this work. Although there exists a hierarchy between product models and between features, we did not consider them but only exploited the hierarchical structure between features and feature values. Exploiting more hierarchical structures will help us model topics more robustly. For example, there may be several products which belong to the same manufacturer, and leveraging such hierarchy can help us avoid bias in data since we can aggregate data for manufacturers as we need. As discussed in the previous sections, there exist other limitations. For example, the product specific topics are not robust when there are not enough review texts for products. Also, as we employ bag-of-words representation, the prior words in specifications may attract words that are not really about the features or feature values in the specifications.

While we retrieve all sentences relevant to feature values, one can also suggest retrieving sentences according to sentiment. Retrieving positive, negative, and neutral sentences separately for feature values will help consumers understand the features in a more organized way. Our model also does not consider the time of product reviews being written. A customer feedback on a feature value may be different depending on the time the customer uses it, so disregarding time may result in inconsistency of sentiment on the feature value. If a model considers time, then we can also see how people's opinions change over time and can predict which feature value will be preferred, which may be highly informative for manufacturers. We leave all these interesting problems as future work.

Chapter 6

Retrieval of Relevant Opinion Sentences for New Products[1]

6.1 Introduction

In the previous chapter, we discussed how we can augment product specifications so that users can obtain useful information that are needed to make purchase decisions. Once a user understands details of a product, the user may look for other users' opinions on the product, in order to gain indirect experience. Luckily, online retailers often provide a space on their web sites where users can leave their opinions on each product. However, the majority of products does not have user reviews since they are new or unpopular. Therefore, in this chapter, we study how to retrieve relevant opinions for such products from reviews of other products.

The role of product reviews has been more and more important. Reevoo, a social commerce solutions provider, surveyed 1,000 consumers on shopping habits and found that 88 percent of them sometimes or always consult customer reviews before purchase.[2] According to the survey, 60 percent of them said that they were more likely to purchase from a site that has customer reviews. Also, they considered customer reviews more influential (48%) than advertising (24%) or recommendations from sales assistants (22%). With the development of Internet and E-commerce, people's shopping habits have changed, and we need to take a closer look at it in order to provide the best shopping environment to consumers.

Even though product reviews are considered important to consumers, the majority of the products has only a few or no reviews. Products that are not released yet or newly released generally do not have enough reviews. Also, unpopular products in the market lack reviews because they are not sold and exposed to consumers enough. How can we help consumers who are interested in buying products with no reviews? In this work, we propose novel methods to automatically retrieve review text for such products based on reviews of other products. Our key insight is that opinions on similar products may be applicable to the product that lacks reviews. For example, if products X and Y have the same CPU clock rate, then people's opinion on CPU clock rate for product X may be applicable to that for product Y as well. The similarity between products can be computed based on product specifications which are often available, where an example of

[1] Part of this work has been published in [77].
[2] https://www.reevoo.com/news/half-of-consumers-find-social-content-useful-when-shopping-online/

product specifications is shown in Figure 6.1.

Feature	Value
AE/AF Control	FlexiZone
Face Detection	Automatic Face Tracking technology
Digital Video Format	MOV
Image Recording Format	RAW + JPEG
Max Video Resolution	1920 x 1080
AV Interfaces	composite video/audio
Manufacturer	Canon

Figure 6.1: A part of product specifications at CNET.com.

Here is an example of review text we manually retrieved for a certain product's specification "Resolution: 12.1 megapixels" from real reviews of products that have the same resolution.

> 12.1 MP captures very minute details even at highest zoom. This 12.1 megapixel megazoom offers an awesome value as the pictures it produces are on par with some cheap DSLRs. I will not longer bring my big DSR camera on my vacations. 12MP is too much, I use it with 8MP - that's more than plenty. What I like most about the W200 is my ability to get crystal clear 4000x3000 12.1 meg photos without having to spend a couple of thousand dollars on an digital SLR camera body and then even more cash on the accessories (e.g. lens).

Even though these sentences are not necessarily coherent opinions, they are clearly very useful for users to understand the product features and get access to relevant discussions of other users. Since a user would hardly have a clue about opinions on a new product, such a retrieved review text can be expected to be useful. As a minimum, it can be very useful to help users prioritizing what to read in the existing reviews of other products. Not only from a consumer's perspective, but also from a manufacturer's perspective, such techniques would be beneficial to collect opinions on its new or unpopular products. From the retrieved opinions, the manufacturers would be able to predict what consumers would think even before they release their product, and they can react to the predicted feedback in advance.

This work makes the following contributions:

1. We introduce and study a novel problem of relevant opinion retrieval for products that do not have reviews in order to provide useful information to consumers and manufacturers. To the best of our knowledge, no previous work has addressed this problem.

2. To solve the problem, we propose new probabilistic retrieval methods, which are Translation model, Specifications Generation model, and Review and Specifications Generation model, as well as a standard summarization method MEAD and its modified version MEAD-SIM, and a standard ad-hoc retrieval method. Our suggested probabilistic methods are also able to retrieve per-feature opinions for a query product.

3. We create a new data set for evaluating the new problem and conduct experiments to show that our translation model indeed retrieves useful opinions and outperforms other baseline models. We also provide an automatic method to evaluate retrieved sentences for new products.

In order to evaluate the automatically retrieved opinions for a new or unpopular product, we pretend that the query product does not have reviews and predict its review text based on similar products. Then, we compare the predicted review text with the query product's actual reviews to evaluate the performance of suggested methods. Experiment results show that our translation model effectively retrieves opinions for a product that does not have reviews and it significantly outperforms baseline methods.

6.2 Related Works

Reviews are one of the most popular sources in opinion analysis. Opinion retrieval and summarization techniques attracted a lot of attentions because of its usefulness in Web 2.0 environment. There are several surveys which summarize the existing opinion mining work [45, 76, 60]. Compared to text data in other general retrieval problems, opinionated articles such as product reviews have some different characteristics. In opinion analysis, analyzing polarities of input opinions are crucial. Also, majority of the opinion retrieval works are based on product feature (aspect) analysis. They first find sub-topics (features) of a target and show positive and negative opinions for each aspect. By further segmenting the input texts into the smaller units, they showed more details in a structured way [36, 61, 64, 69, 85, 97, 46]. Meanwhile, product reviews have been also employed to predict ratings [75, 29] or sales [20] of a product. However, no existing work addressed the problem of retrieving opinion sentences for new products yet.

In this work, we also utilize unique characteristics of product data: specifications (structured data) as well as reviews (unstructured data). Although product specifications have been provided in many e-commerce web sites, there are only a limited number of studies that utilized specifications for product review analysis.

Zhou and Chaovalit [117] performed sentiment classification on reviews using domain ontology database, which may be regarded as product specifications. Bhattattacharya *et al.* [7] employed IMDb's structured data to categorize documents, and Yu *et al.* [112] built an aspect hierarchy using product specifications and reviews. Wang *et al.* [103] and Peñalver-Martínez *et al.* [82] also employed product specifications to summarize product features. Product reviews and specifications were jointly modeled using topic models by Duan et al. [22] to improve product search and by Park et al. [79] to generate augmented specifications with useful information. Park et al. [79] retrieved review sentences for each (feature, value) pair, but they did not study their model's performance on products with no reviews. In addition, their model does not consider similarity among products or specifications, which is an important factor for the problem. Likewise, there are a few studies that employed product specifications, but their goals are different from ours.

Our work is related to text summarization, which considers centrality of text. Automatic text summarization techniques have been studied for a long time due to the need of handling large amount of electronic text data [74, 48, 35]. Automatic summarization techniques can be categorized into two types, extractive summarization and abstractive summarization. Extractive summarization makes a summary by selecting representative text segments, usually sentences, from the original documents. Abstractive summarization does not directly reuse the existing sentences but generates sentences based on text analysis. Our work is similar to extractive summarization in that we select sentences from original documents but different in that we retrieve sentences for an entity that does not have any text. Among the previous work, MEAD [86] is one of the most popular public extractive summarization toolkits, which supports multi-document summarization in general domain. The goal of MEAD is different from ours in that we want a summary for a specific product, and also, MEAD does not utilize external structured data (specifications).

Cold start problem in recommendation systems [90], where no one has rated new items yet, is also related to our problem. However, unlike rating connections between items and users, each user review carries its unique and complex meaning, which makes the problem more challenging. Moreover, our goal is to provide useful relevant opinions about a product, not recommending a product. XML retrieval [49] that utilizes structured information of documents is also related to our work, in that reviews and specifications can be represented as a special XML. However, unlike general XML retrieval, in this work, we propose more specialized methods for product reviews using product category and specifications. In addition, because we require the retrieved sentences to be central in reviews, we consider both centrality and relevance while general retrieval methods focus on relevance only. As far as we know, none of the existing work tried to solve the same problem as ours.

6.3 Problem Definition

The product data consist of N products $\{P_1, ..., P_N\}$. Each product P_i consists of its set of reviews $R_i = \{r_1, ..., r_m\}$ and its set of specifications $S_i = \{s_{i,1}, ..., s_{i,F}\}$, where a specification $s_{i,k}$ is a feature-value pair, $(f_k, v_{i,k})$, and F is the number of features. Given a query product P_z, for which R_z is not available, our goal is to retrieve a sequence of relevant opinion sentences T in K words for P_z. Figure 6.2 illustrates our novel problem.

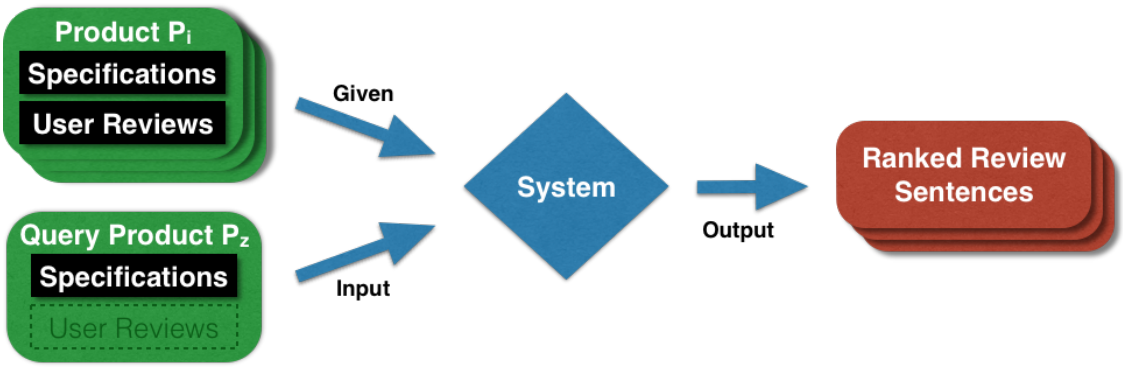


Figure 6.2: Relevant opinion retrieval for query products. Note that the query products do not have any user reviews.

Note that our problem setup is a mixture of retrieval and summarization. On the one hand, it can be regarded as a ranking problem, similar to retrieval; on the other hand, it can also be regarded as a summarization problem since we restrict the total number of words in the retrieved opinions .

This is a new problem that has not been addressed in any previous work. The problem is challenging for several reasons. Retrieved sentences for P_z should conform to its specifications S_z while we do not know which sentences are about which specific feature-value pair. In addition, the retrieved sentences should be central across relevant reviews so that they reflect central opinions. Despite the challenges, we try to show that achieving the goal is feasible. In the next sections, we propose multiple methods to solve the problem.

6.4 Overall Approach

When reviews are not available for a product, a consumer has no way to obtain opinions on it. In order to help consumers in such situation, we believe that product specifications are the most valuable source to find similar products. We thus leverage product specifications to find similar products and choose relevant sentences from their user reviews. In this approach, we assume that if products have similar specifications, the reviews are similar as well. For example, here is an actual review sentence from the review of a digital

camera that takes a picture at high resolution: "the best camera I have ever owned, takes unbelievable crisp sharp photos with it's 16.1 Megapixels." It is admitted that the consumer is very impressed with the feature-value pair, ("Resolution", "16.1 Megapixels"), and we can expect that other digital cameras with the same feature-value pair could impress their consumers as well. The assumption may not be valid in some cases, *i.e.*, the same specifications may yield very different user reviews. We thus try to retrieve "central" opinions from similar products so that the retrieved sentences can become clearly useful.

6.5 Similarity between Products

We assume that similar products have similar feature-value pairs (specifications). In general, there are many ways to define a similarity function. We are interested in finding how well a basic similarity function will work although our framework can obviously accommodate any other similarity functions. Therefore, we simply define the similarity function between products as

$$SIM_p(P_i, P_j) = \frac{\sum_{k=1}^{F} w_k SIM_f(s_{i,k}, s_{j,k})}{\sum_{k=1}^{F} w_k} \tag{6.1}$$

where w_k is a weight for the feature f_k, and the weights $\{w_1, ..., w_F\}$ are assumed identical ($w_k = 1$) in this study, so the similarity function becomes

$$SIM_p(P_i, P_j) = \frac{\sum_{k=1}^{F} SIM_f(s_{i,k}, s_{j,k})}{F} \tag{6.2}$$

where $SIM_f(s_{i,k}, s_{j,k})$ is a cosine similarity for feature f_k between P_i and P_j and is defined as

$$SIM_f(s_{i,k}, s_{j,k}) = \frac{\mathbf{v_{i,k}} \cdot \mathbf{v_{j,k}}}{\sqrt{\sum_{v \in \mathbf{v_{i,k}}} v^2} \sqrt{\sum_{v \in \mathbf{v_{j,k}}} v^2}} \tag{6.3}$$

where $\mathbf{v_{i,k}}$ and $\mathbf{v_{j,k}}$ are phrase vectors in values $v_{i,k}$ and $v_{j,k}$, respectively. Both $SIM_p(P_i, P_j)$ and $SIM_f(s_{i,k}, s_{j,k})$ range from 0 to 1.

In this work, we define the phrases as comma-delimited feature values. $SIM_f(s_{i,k}, s_{j,k})$ is similar to cosine similarity function, which is used often for measuring document similarity in Information Retrieval (IR), but the difference is that we use a phrase as a basic unit while a word unit is usually adopted in IR. We use a phrase as a basic unit because majority of the words may overlap in two very different feature values. For example, the specification phrases "Memory Stick Duo", "Memory Stick PRO-HG Duo", "Memory Stick PRO Duo", and "Memory Stick PRO Duo Mark2" have high word cosine similarities among themselves since

they at least have 3 common words while the performances of the specifications are very different. Thus, our similarity function with phrase unit counts a match only if the phrases are the same.

6.6 Methods

In this section, we suggest multiple methods for relevant opinion sentences retrieval. We first suggest a standard summarization tool, MEAD [86]. In order to make up for the MEAD's weak points, we also suggest modified version of MEAD. Then, we propose our probabilistic models to solve the problem.

6.6.1 MEAD: Retrieval by Centroid

For our problem, text retrieval based only on query-relevance is not desirable. The retrieved sentences need to be central in other reviews in order to obtain central opinions about specifications. For example, if there are more opinions that contains a word "big" than a word "small" for a certain feature-value pair, it is desired to assign higher score to the sentences having the word "big". However, since the query contains only feature-value pair words, classic information retrieval approaches are not able to prefer such sentences. Therefore, we suggest using a method that considers centrality among sentences.

MEAD [86] is a popular centroid-based summarization tool for multiple documents, and it was shown to effectively generate summaries from a large corpus. It provides an auto-generated summary for multiple documents. For a corpus R, a score of ith sentence t in a document is computed by sum of centroid and position scores of words, which is defined as

$$score(t; R) = w_c C_t + w_o O_t \tag{6.4}$$

where C_t is a sum of centroid scores of words in t, which is defined as $C_t = \sum_w C_{w,t}$, and O_t is a position score, which gives higher score to the sentences appearing earlier in a document and defined as $O_t = \frac{(n-i+1)}{n} \cdot C_{max}$ where n is the number of sentences in the document and C_{max} is the maximum centroid score in the document. Centroid score of a word, $C_{w,t}$, is a TFIDF value in the corpus R, and w_c and w_o are weights for C_t and O_t, respectively. Please refer to [86] for more details.

In order to retrieve sentences that are likely to be relevant to the query product P_z, which has no reviews, we employ specifications to find products similar to P_z and use the similarity as a clue for finding relevant sentences. Since the score formula (6.4) utilizes only centrality and does not consider relevance to the query product, we augment it with product similarity to P_z so that we can find sentences that are query-relevant and central at the same time. In addition, MEAD employs position score that is reasonable for news articles,

but it may not be appropriate for reviews; unlike news articles, it is hard to say that the sentences appearing earlier in the reviews are more important than those appearing later. Thus, we remove position score term from formula (6.4), and we augment it with similarity to query. The new score function is defined as

$$score(t, S_y; R, S_z) = C_t \cdot SIM_p(S_y, S_z) \tag{6.5}$$

where t is a sentence in a review for product P_y and $SIM_p\ (S_y, S_z)$ is a product similarity between P_y and the query product P_z, which is defined in equation (6.2).

6.6.2 Probabilistic Retrieval

To solve the problem in a more principled way, we introduce our probabilistic methods. Query likelihood retrieval model [6], which assumes that a document generates a query, has been shown to work well for ad-hoc information retrieval. Similarly, we attempt to generate the query specifications S_z from a candidate sentence t via several generative scenarios.

Specifications Generation Model

The generative story is described as follows. Each sentence t from reviews of its product P_y first generates its specifications S_y. The specifications S_y then generate the query specifications S_z. Following the dependencies among variables, the scoring function is defined as

$$\begin{aligned} score(t, S_y; R, S_z) &\propto p(t, S_y|S_z) \\ &= \frac{p(S_z|S_y)p(S_y|t)p(t)}{p(S_z)} \end{aligned} \tag{6.6}$$

We can interpret $p(t, S_y|S_z)$ as the probability that t and S_y satisfy information needs of a user given S_z. $p(S_z|S_y)$ measures proximity of S_y to S_z. $p(S_y|t)$ measures proximity of t to S_y, and $p(t)$ is a general preference on t. Since we assume no preference on sentences, we ignore $p(t)$ for ranking. $p(S_z)$ is also ignored because it does not affect the ranking of sentences for S_z. Thus, the formula assigns high score to a sentence if its specifications S_y match S_z well and the sentence t matches its specifications S_y well. $p(t, S_y|S_z)$ is then defined as

$$\begin{aligned} p(t, S_y|S_z) &\propto p(S_z|S_y)p(S_y|t) \\ &= \sum_{k=1}^{F} p(s_{z,k}|s_{y,k})p(s_{y,k}|t) \end{aligned} \tag{6.7}$$

where a set of specifications such as S_y is decomposed into feature-value pairs $s_{y,k}$. We assume that a k'th feature-value pair of one specification set generates only the k'th feature-value pair of another specification set, not other feature-value pairs. This is to ensure that sentences not related to a specification $s_{z,k}$ are scored low even if their word score $p(s_{y,k}|t)$ is high. $p(s_{z,k}|s_{y,k})$, proximity of $s_{y,k}$ to $s_{z,k}$, is estimated as follows.

$$p(s_{z,k}|s_{y,k}) \propto \frac{SIM_f(s_{z,k}, s_{y,k})}{\sum_{s \in Distinct(k)} SIM_f(s, s_{y,k})} \tag{6.8}$$

where $Distinct(k)$ is a set of distinct feature-value pairs for a feature f_k. $p(s_{y,k}|t)$ is defined as

$$p(s_{y,k}|t) = \prod_{w \in s_{y,k}} p(w|t) = \prod_{w \in U} p(w|t)^{c(w,s_{y,k})} \tag{6.9}$$

where U is a vocabulary set in corpus R, and $c(w, s_{y,k})$ is a count of word w in the feature-value pair $s_{y,k}$. $p(w|t)$ follows t's unigram language model [84], and it means a word w's likelihood in a sentence t. One of the standard ways to estimate $p(w|t)$ is using maximum likelihood (ML) estimator, which gives $p(w|t) = \frac{c(w,t)}{|t|}$, where $c(w,t)$ is the count of w in t, and $|t|$ is the number of words in t. Thus, $p(s_{y,k}|t)$, likelihood of a feature-value pair $s_{y,k}$ in a sentence t, becomes higher if more words in the feature-value pair appear often in t. To avoid over-fitting and prevent $p(s_{y,k}|t)$ from being zero, we smooth $p(w|t)$ with Jelinek-Mercer smoothing method [41], which is shown in [114] to work reasonably well. Using Jelinek-Mercer smoothing, $p(w|t)$ is defined as

$$p(w|t) = (1 - \lambda)p_{ml}(w|t) + \lambda p(w|R) \tag{6.10}$$

where $p_{ml}(w|t)$ and $p(w|R)$ follow a sentence language model and a corpus language model, respectively, which are estimated with ML estimator. To smooth $p(w|t)$, a reference language model $p(w|R)$ is used so that we can have general word likelihood that nicely augments $p_{ml}(w|t)$. The resulting $p(w|t)$ can be regarded as weighted average of $p_{ml}(w|t)$ and $p(w|R)$.

Review and Specifications Generation Model

Specifications Generation model in section 6.6.2 does not consider centrality among reviews. However, as explained in section 6.6.1, centrality as well as query-relevance should be considered for the task. Here, we assume that a candidate sentence t of product P_y generates the product's reviews R_y except itself t. This generation enables us to measure centrality of t among all other sentences in the reviews for P_y. Then, t and $R_y^{\setminus t}$ jointly generate its specifications S_y, where $R_y^{\setminus t}$ is a set of reviews for P_y except the sentence t.

Intuitively, it makes more sense for S_y to be generated by both t and $R_y^{\setminus t}$ than by only t. S_y then generates the query specifications S_z. Following the dependencies, the score function is defined as

$$\begin{aligned} score(t, R_y^{\setminus t}, S_y; R, S_z) &\propto p(t, R_y^{\setminus t}, S_y | S_z) \\ &= \frac{p(S_z|S_y)p(S_y|t, R_y^{\setminus t})p(R_y^{\setminus t}|t)p(t)}{p(S_z)} \\ &\propto p(S_z|S_y)p(S_y|t, R_y^{\setminus t})p(R_y^{\setminus t}|t) \end{aligned} \tag{6.11}$$

where $p(t)$ and $p(S_z)$ are ignored for the same reason as in section 6.6.2. Now, $p(R_y^{\setminus t}|t)$, a proximity of t to the reviews $R_y^{\setminus t}$, is computed to consider centrality of t. Also, $p(S_y|t, R_y^{\setminus t})$, a proximity of t and $R_y^{\setminus t}$ to the specifications S_y, is computed to promote sentences from reviews that match its specifications well. Thus, a sentence t is preferred if (1) its specifications S_y is similar to S_z, (2) S_y represents its reviews R_y well, and (3) $R_y^{\setminus t}$ represents t well. $p(t, R_y^{\setminus t}, S_y|S_z)$ can be re-written as

$$p(t, R_y^{\setminus t}, S_y|S_z) = p(R_y^{\setminus t}|t) \sum_{k=1}^{F} p(s_{z,k}|s_{y,k}) \prod_{w \in s_{y,k}} p(w|t, R_y^{\setminus t}) \tag{6.12}$$

where $p(w|t, R_y^{\setminus t})$ is smoothed to $(1-\lambda)\delta(w|t, R_y^{\setminus t}) + \lambda p(w|R)$, where $\delta(w|t, R_y^{\setminus t})$ is defined as

$$\delta(w|t, R_i) = \begin{cases} 0 & \text{if } w \notin t \\ \frac{c(w,t)+c(w,R_i)}{|t|+|R_i|} & \text{if } w \in t \end{cases} \tag{6.13}$$

We ignore w if w is not in t in order to require the retrieved sentences to contain words in $s_{y,k}$. The proximity of t to $R_y^{\setminus t}$, $p(R_y^{\setminus t}|t)$, is estimated by TFIDF cosine similarity function $SIM(R_y^{\setminus t}, t)$, where TFIDF cosine similarity between documents d and d' is defined as

$$SIM(d, d') = \frac{\sum_{w \in d, d'} c(w, d) \cdot c(w, d') \cdot IDF(w)^2}{\sqrt{\sum_{w \in d}(c(w, d) \cdot IDF(w))^2} \cdot \sqrt{\sum_{w' \in d'}(c(w', d') \cdot IDF(w'))^2}} \tag{6.14}$$

where IDF of word w is defined as

$$IDF(w) = \log \frac{|R|}{1 + DF(w)} \tag{6.15}$$

where $|R|$ is the number of reviews in the whole corpus, and $DF(w)$ is the number of documents that contain w.

Translation Model

In Review and Specifications Generation model, we assumed a sentence t of product P_y generates its reviews R_y, and t and R_y jointly generate their specifications S_y. However, we can also assume that t generates reviews of an arbitrary product because there may be better reviews that can represent t and generate S_y well. In other words, there may be a product P_x that translates t and generates P_z based on the translation with a better performance.

The generative story is described as follows. A candidate sentence t of a product P_y generates each review set of all products, which will be used as translations of t. t and each of the generated review sets, R_x, jointly generate t's specifications S_y, and S_y generates specifications of R_x, S_x, and the query specifications S_z. We intend S_y to generate specifications of the translating product S_x so as to penalize the translating product if its specifications are not similar to S_y. Following the generative story, the score function is defined as

$$\begin{aligned} score(t, S_y; R, S_z) &\propto p(t, S_y|S_z) \\ &= \frac{p(S_z|S_y) \sum_{P_x \in P^{\setminus z}} p(S_x|S_y) p(S_y|t, R_x) p(R_x|t) p(t)}{p(S_z)} \\ &\propto p(S_z|S_y) \sum_{P_x \in P^{\setminus z}} p(S_x|S_y) p(S_y|t, R_x) p(R_x|t) \end{aligned} \tag{6.16}$$

where $p(S_z)$ and $p(t)$ are ignored for the same reason as before. As described, the score function contains a loop over all products (except P_z), instead of using only t's review set R_y, to get the votes from all translating products. The features in different specifications are paired together, which decompose $p(t, S_y|S_z)$ as follows.

$$\begin{aligned} p(t, S_y|S_z) &\propto \sum_{k=1}^{F} p(s_{z,k}|s_{y,k}) \sum_{P_x \in P^{\setminus z}} p(s_{x,k}|s_{y,k}) p(s_{y,k}|t, R_x) p(R_x|t) \\ &= \sum_{k=1}^{F} p(s_{z,k}|s_{y,k}) \sum_{P_x \in P^{\setminus z}} p(s_{x,k}|s_{y,k}) p(R_x|t) \prod_{w \in s_{y,k}} p(w|t, R_x) \end{aligned} \tag{6.17}$$

where proximity between specifications are estimated using cosine similarity function SIM_f as in specifications generation model, and the proximity of t to arbitrary reviews R_x, $p(R_x|t)$, is estimated by TFIDF cosine similarity function. In order to consider the case P_y is the same as P_x, we define $p(w|t, R_x)$ as

$$p(w|t, R_x) = \begin{cases} (1-\lambda)\delta(w|t, R_x) + \lambda p(w|R) & \text{if } P_x \neq P_y \\ (1-\lambda)\delta(w|t, R_x^{\setminus t}) + \lambda p(w|R) & \text{if } P_x = P_y \end{cases} \tag{6.18}$$

Meanwhile, looping over all non-query products is probably too expensive in terms of computational complexity. We thus choose X translating products P^X to reduce the complexity. Perhaps, the most

promising translating products may be those who are similar to the query product P_z. We want the retrieved sentences to be translated well by the actual reviews of P_z, which means that those reviews of products not similar to P_z are not considered important. Since we assume that products similar to P_z are likely to have similar reviews, we exploit the similar products' reviews to approximate R_z, where we measure similarity using specifications. Therefore, we loop over only X translating products P^X that are most similar to P_z, where similarity function SIM_p is employed to measure similarity between products. Since P_x needs to be similar to P_z, we further assume that P_x generates P_z, which yields proximity of P_x to P_z, $p(P_z|P_x)$, and it is defined as

$$p(P_z|P_x) = \frac{SIM_p(P_z, P_x)}{\sum_{x' \in P^X} SIM_p(P_z, P_{x'})} \tag{6.19}$$

and this product-level similarity is used as a weight of P_x in formula (6.17).

6.7 Experimental Setup

6.7.1 Data Set

Since we study a new task that has not been studied before, there is no existing test collection available to use for evaluation. We thus must solve the challenge of creating a test set. We address this problem by using products with known reviews as test cases. We pretend that we do not know their reviews and use our methods to retrieve sentences in K words; we then compare the retrieved text with the actual reviews of a test product. This allows us to evaluate the task without manual annotation, and it is a reasonable way to perform evaluation because it would reward a system that can retrieve review sentences that are similar to the actual review sentences of a product.

We now describe how to build our data set in detail. First, it is required for our problem to obtain reviews and specifications for products, and these kinds of data are available in several web sites such as Amazon.com, BestBuy.com, and CNET.com. Among them, we chose CNET.com because they have a reasonable amount of reviews and relatively well-organized specifications. There are several product categories in CNET.com, and we chose digital camera and MP3 player categories since they are reasonably popular and therefore the experiment results can yield significant impact. From CNET.com, we crawled product information for all products that were available on February 22, 2012 in both categories. For each product, we collected all of its user reviews and specifications.

We omitted products that do not contain reviews or specifications. (We found that about two thirds of

Table 6.1: Statistics of the data for digital camera and MP3 player categories.

	Digital Camera	MP3 Player
Num. of products	1,153	605
Num. of reviews	12,779	14,159
Num. of sentences	137,599	291,858
Num. of word tokens	754,888	1,725,192
Vocabulary size	6,442	6,959
Num. of features	9	8
Num. of distinct feature values	1,038	384

the products indeed didn't have any user reviews.) To preprocess the review text, we performed sentence segmentation, word tokenization, and lemmatization using Stanford CoreNLP [66] version 1.3.5. We lowered word tokens and removed punctuation. Then, we removed word tokens that appear in less than five reviews and stopwords.

We also preprocessed specifications data in the following way. In general, specifications contain dozens or hundreds of distinct features, and many of them are not mentioned in the reviews. Therefore, we chose features that are considered important by users. In order to choose such key features, we simply adopted highlighted features provided by CNET.com assuming that they chose the features based on importance. The highlighted features are listed in Table 6.2. We removed feature values that appear in less than five products. Then, we tokenized the feature and feature value words, and we lowered the word tokens. The statistics of the reviews and specifications data is shown in Table 6.1. While digital camera category has more products, more reviews are written for mp3 player categories. Also, in general, users wrote more texts per review for mp3 players than digital cameras. The number of highlighted features used for digital cameras is similar to that for mp3 players while there are much more distinct feature values for digital cameras.

Table 6.2: CNET.com's highlighted features for digital camera and MP3 player categories.

Digital Camera	MP3 Player
Manufacturer	Manufacturer
Product Type	Product Type
Resolution	Digital Storage
Digital Video Format	Flash Memory Installed
Image Stabilizer	Built-in Display – Diagonal Size
Lens System – Type	Battery / Power – Battery
Memory / Storage – Supported Mem. Cards	Digital Player / Recorder – Supported Digital Audio Standards
Camera Flash – Camera Flash	Battery / Power – Mfr. Estimated Battery Life
Optical Sensor Type	

In order to evaluate the performance of our methods for retrieving review sentences for a new or unpopular

product, we perform the following experiment. To choose test products, which will be regarded as products with no reviews, we selected top 50 products by the number of reviews in each category in order to obtain reliable gold standard data. Please note that we did not select products that have their different versions such as colors or editions, in order to ensure that the candidate products do not have the same reviews as the test products. For each test product, P_z, all sentences of other products are regarded as candidate sentences. Pretending that P_z does not have any reviews, we rank those candidate sentences and generate a text with the first K word tokens, and we compare it with the actual reviews of P_z. We assume that if the generated review text is similar to the actual reviews, it is a good review text for P_z. The average number of reviews in the top 50 products is 78.5 and 152.2 for digital cameras and mp3 players, respectively. For the probabilistic retrieval models, we use λ to control the amount of smoothing for language models, and we empirically set it to 0.5 for both product categories, which showed the best performance.

6.7.2 Evaluation Metrics

To evaluate a quality of the length-K retrieved text based on actual reviews for the query, we face another challenge: how should we measure the performance? We could consider using standard retrieval measures, but neither NDCG nor MAP seems appropriate since we do not have multiple levels of judgments or even binary judgments. We thus decided to measure the proximity between the retrieved text and the actual reviews. Regarding the retrieved text as a summary for the query product, we can view our task as similar to multiple document summarization, whose goal is to generate a summary of multiple documents. Thus, we employ ROUGE evaluation method [57], which is a standard evaluation system for multiple document summarization. In general, ROUGE evaluates the quality of an automatically generated summary by comparing it with one or more manually generated reference summaries. Assuming the actual reviews of the query product are manually generated reference summaries, we can adopt ROUGE to evaluate the retrieved sentences. Among various ROUGE metrics, we employ ROUGE-1 and ROUGE-2, which are unigram and bigram matching metrics, respectively, and they have been shown to perform well for the task. We compute precision, recall, and F1-score of each metric. For example, precision of ROUGE-n is defined as

$$\text{ROUGE-}n(r,s) = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)} \tag{6.20}$$

where r and s are reference and retrieved summaries, respectively, $gram_n$ is n-gram text, $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in the retrieved summary and a reference summary. When there are multiple reference summaries available, they use the following evaluation formula.

$$\text{ROUGE-}n_{multi} = max_i \text{ROUGE-}n(r_i, s) \tag{6.21}$$

Please note that each of the precision, recall, and F1-score takes the maximum from the reference summaries. More details about ROUGE can be found in [57].

However, the problem of ROUGE metrics is that it does not consider importance of words. All words have different level of importance; for example, a word such as "of" is much less important than a word "megapixel" since "of" appears too often in documents and does not carry useful information. If a retrieved text contains many unimportant words, it may obtain a high score by ROUGE metrics, which is not desired. Therefore, we also employ TFIDF cosine similarity, which considers word importance by Inverse Document Frequency (IDF). TFIDF cosine similarity function between two documents is defined in equation (6.14). While the formula measures similarity based on bag of words, bigram provides important information about distance among words, so we adopt bigram-based TFIDF cosine similarity as well. Similar to ROUGE-n_{multi}, we take a maximum from $SIM(r_i, s)$ among different reference summaries because we still evaluate based on multiple reference summaries. For both ROUGE and SIM metrics, we use retrieved text length 100, 200, and 400, which reflect diverse users' information needs.

6.8 Experiment Results

6.8.1 Qualitative Analysis

Table 6.3: Top ten sentences retrieved for Pentax *ist DS (Digital Camera) by Translation model with X=5.

(1) This was my first and my last Pentax .
(2) This pentax is a great value for money , and a nice entry level dslr , compatible with most Pentax lens .
(3) I have found the Pentax DL to be high quality , with great features .
(4) Nice job pentax .
(5) I have been a Pentax SLR user for years , beginning with the SuperProgram , ZX-50 , and ZX-5n .
(6) When I bought it , I was in bankruptcy and the cheaper Pentax came to me .
(7) Pentax have been making great lenses and cameras for a long time , and this range is no exception .
(8) Great photos , color , ease of use , compact size , compatible with Pentax mount lenses .
(9) I had owned a great 35mm Pentax camera before that took wonderful pictures , which , after 20 years went caput .
(10) Very smart Pentax .

Table 6.4: Specifications for Pentax *ist DS. Note that some feature values are not available.

Feature	Value
Manufacturer	Pentax
Product Type	Digital camera - SLR
Resolution	6.1 megapixels
Digital Video Format	
Image Stabilizer	
Lens System – Type	3 x x Zoom lens - 18 mm - 55 mm - F/3.5-5.6 DA Pentax KAF
Memory / Storage – Supported Mem. Cards	SD Memory Card
Camera Flash – Camera Flash	Pop-up flash
Optical Sensor Type	CCD

In order to see the usefulness of the sentences retrieved by our novel Translation model, we show the top retrieved sentences for query products and compare them with the actual review sentences for the query products. Table 6.3 lists top retrieved sentences for a product in each category, where the sentences are ordered by their scores, and the specifications of the product is listed in Table 6.4.We set the number of translating products to five, which is reasonable if we consider the computational cost of the model.

For the digital camera Pentax *ist DS, several top retrieved sentences such as (2) and (8) mention about its compatibility with Pentax lenses. Surprisingly, there were several reviews for Pentax *ist DS that praise its lens compatibility, and here are two actual examples from review sentences: "Plus the DS is backwards compatible with all old Pentax lenses, which have a well-deserved reputation among photographers." "I can use my pile of old (and very old) Pentax lenses including the m42 lenses." Also, the retrieved sentences such as (7), (8), (9), and possibly (3) mention about Pentax's great picture quality, which is supported by the following actual review sentences: "Amazingly sharp lens." "It has a much better lens package than the Rebel and the base 20D kit." Sentences (2) and (6) claim the product's good value, which is again supported by actual review sentences: "Better value than you think" "The camera is also cheaper than the comparable Nikon and Canon." The retrieved sentence such as (8) mentions about ease of use for the camera, and many users actually complimented the camera on its ease of use, indeed. The supporting sentences are as follows: "Very easy to use right out of the box." "The controls are very easy to learn and are, for the most part, very intuitive." Meanwhile, the sentence (1) carries inconsistent opinion, which shows negative sentiment on Pentax camera. Nevertheless, in a user's perspective, who does not know much about Pentax *ist DS or other Pentax cameras, the listed information would be highly informative especially if the camera has no or few reviews. Although some of the retrieved sentences do not carry useful information, it is clear that some other retrieved sentences are indeed useful.

Our probabilistic retrieval models have a capability of retrieving relevant sentences for a specific feature. For each of the probabilistic models, we can assume that the number of features F is one so that the score functions compute only for one feature. Table 6.5 shows top retrieved sentences for the feature "Lens System – Type" of Pentax *ist DS. As found in the top sentences for the whole product in Table 6.3, we can easily find that all the sentences except (2) praise the lens compatibility of Pentax, indeed. In addition, all sentences except (1) praise high quality of its lens, which is coherent with the top sentences for the whole product. From the sentences, users can learn much about the given product's lens such as other consumers' general sentiment and specific reasons why they like or dislike its lens.

Table 6.5: Top sentences retrieved by Translation model (X=5) specifically for the feature "Lens System – Type" of Pentax *ist DS.

(1) This pentax is a great value for money , and a nice entry level dslr , compatible with most Pentax lens .
(2) Pentax have been making great lenses and cameras for a long time , and this range is no exception .
(3) Great photos , color , ease of use , compact size , compatible with Pentax mount lenses .
(4) The kit lens is better than what ships with some competitors , and the camera is compatible with most older Pentax lenses , making it possible to save hundreds by buying used lenses rather than having to sink money into new digital lenses .
(5) Compatibility with older Pentax lenses is a real bonus too , as these are usually of very high quality and can be picked up at good prices second-hand .

Manually finding relevant opinions for a query product or its specific feature is extremely time-consuming for users; they need to find similar products by manually comparing specifications and extract relevant and central sentences from all the reviews of the similar products, which may take too much time. Here, we verified the automatically retrieved sentences can be indeed useful for users. In the next section, we quantitatively compare our Translation model with other suggested methods.

6.8.2 Quantitative Evaluation

To retrieve review sentences that are likely to be written for a new or unpopular product, we employ several methods. In order to see the effectiveness of a standard ad-hoc retrieval method, we employ query likelihood (QL) language model approach [84], and we define the score function as $score(t; R, S_z) = \sum_{k=1}^{F} \prod_{w \in s_{z,k}} p(w|t)$, where $p(w|t)$ is smoothed as in equation (6.10). We suggested a modified version of one of the standard summarization tools, MEAD-SIM in formula (6.5), which considers both query-relevance and centrality. We employ MEAD-SIM as one of the baseline methods, and we also show results from the basic

Table 6.6: Unigram and bigram TFIDF cosine similarity @ K for Digital Camera and MP3 Player categories.

Category	Model	COS1@100	COS1@200	COS1@400	COS2@100	COS2@200	COS2@400
Digital	QL	0.112	0.128	0.147	0.0185	0.0215	0.0257
Camera	MEAD	0.131	0.124	0.141	0.0258	0.0204	0.0184
	MEAD-SIM	0.136	0.130	0.158	0.0271	0.0226	0.0223
	SpecGen	0.143	0.173	0.206	0.0230	0.0270	0.0291
	ReviewSpecGen	0.171	0.208	0.231	0.0210	0.0244	0.0298
	Translation	**0.314**†‡	**0.327**†‡	**0.333**†‡	**0.0736**†‡	**0.0743**†‡	**0.0794**†‡
	(increase %)	(+131%)	(+152%)	(+111%)	(+172%)	(+229%)	(+256%)
MP3	QL	0.090	0.99	0.118	0.0147	0.0159	0.0173
Player	MEAD	0.089	0.078	0.091	0.0123	0.0128	0.0117
	MEAD-SIM	0.131	0.136	0.145	0.0206	0.0197	0.0178
	SpecGen	0.153	0.183	0.208	0.0225	0.0274	0.0294
	ReviewSpecGen	0.206	0.227	0.253	0.0261	0.0270	0.0327
	Translation	**0.267**†‡	**0.297**†‡	**0.316**†‡	**0.0458**†‡	**0.0567**†‡	**0.0649**†‡
	(increase %)	(+104%)	(+118%)	(+118%)	(+104%)	(+188%)	(+265%)

MEAD in formula (6.4) to see the effect of query-relevance addition to MEAD; we set $w_c = 1$ and $w_o = 0$ since position score is inappropriate for reviews. We also introduced several probabilistic retrieval methods for the task. Review and Specifications Generation model (ReviewSpecGen) considers both query-relevance and centrality, so we use it as another baseline method. Specifications Generation model (SpecGen) focuses on query-relevance, and we show its results to compare with ReviewSpecGen and QL. We then suggested our novel Translation model (Translation). We tuned X to be 100 for digital cameras and 10 for mp3 players, unless otherwise specified, which showed the best TFIDF cosine similarity values. The results from Translation model are mainly compared with the two baselines MEAD-SIM and ReviewSpecGen. † and ‡ are used to mark if the improvement for Translation model is statistically (paired t-test with p=0.05) significant in each measure from MEAD-SIM and ReviewSpecGen, respectively. We also record how much Translation model improves MEAD-SIM in parentheses.

Table 6.6 shows TFIDF cosine similarity evaluation results for both digital cameras and mp3 players. Both unigram (COS1) and bigram (COS2) measures are listed for the suggested methods. In general, models that exploit specifications as query (MEAD-SIM, SpecGen, ReviewSpecGen, and Translation) except QL outperform MEAD, which does not compute query-relevance. QL outperforms MEAD in mp3 player data set, but it does not outperform other models in both data sets, because it does not consider specifications similarity between products. MEAD-SIM outperforms MEAD in all cosine similarity measures (12/12), which means that centrality alone cannot perform well. ReviewSpecGen adds centrality computation to SpecGen, and the results show that its centrality helps it outperform SpecGen in most measures (9/12). ReviewSpecGen outperforms MEAD-SIM in all unigram measures (6/6) and most bigram measures (5/6). Translation model significantly outperforms MEAD-SIM in all measures (12/12), and the average performance increase percentage is 162%. It also significantly outperforms ReviewSpecGen in all measures (12/12), which means that choosing products similar to the query product as translating products was more effective than choosing only one product the candidate sentence belongs to. Translation model outperforms other models especially in bigram measures, which means that Translation model retrieves more connected fragments that are in the actual reviews.

We also evaluate retrieval results with ROUGE metrics. Although ROUGE does not consider importance of words, it is able to compute recall, precision, and F1 score in both unigram (ROUGE1-R, ROUGE1-P, and ROUGE1-F) and bigram (ROUGE2-R, ROUGE2-P, and ROUGE2-F) units. The ROUGE evaluation results for mp3 players are shown in Table 6.7. QL outperforms MEAD in all measures, but it is outperformed by MEAD-SIM in all measures since QL does not consider specifications similarity between products. SpecGen outperforms ReviewSpecGen in most measures (13/ 18) especially in bigram measures (9/9), which

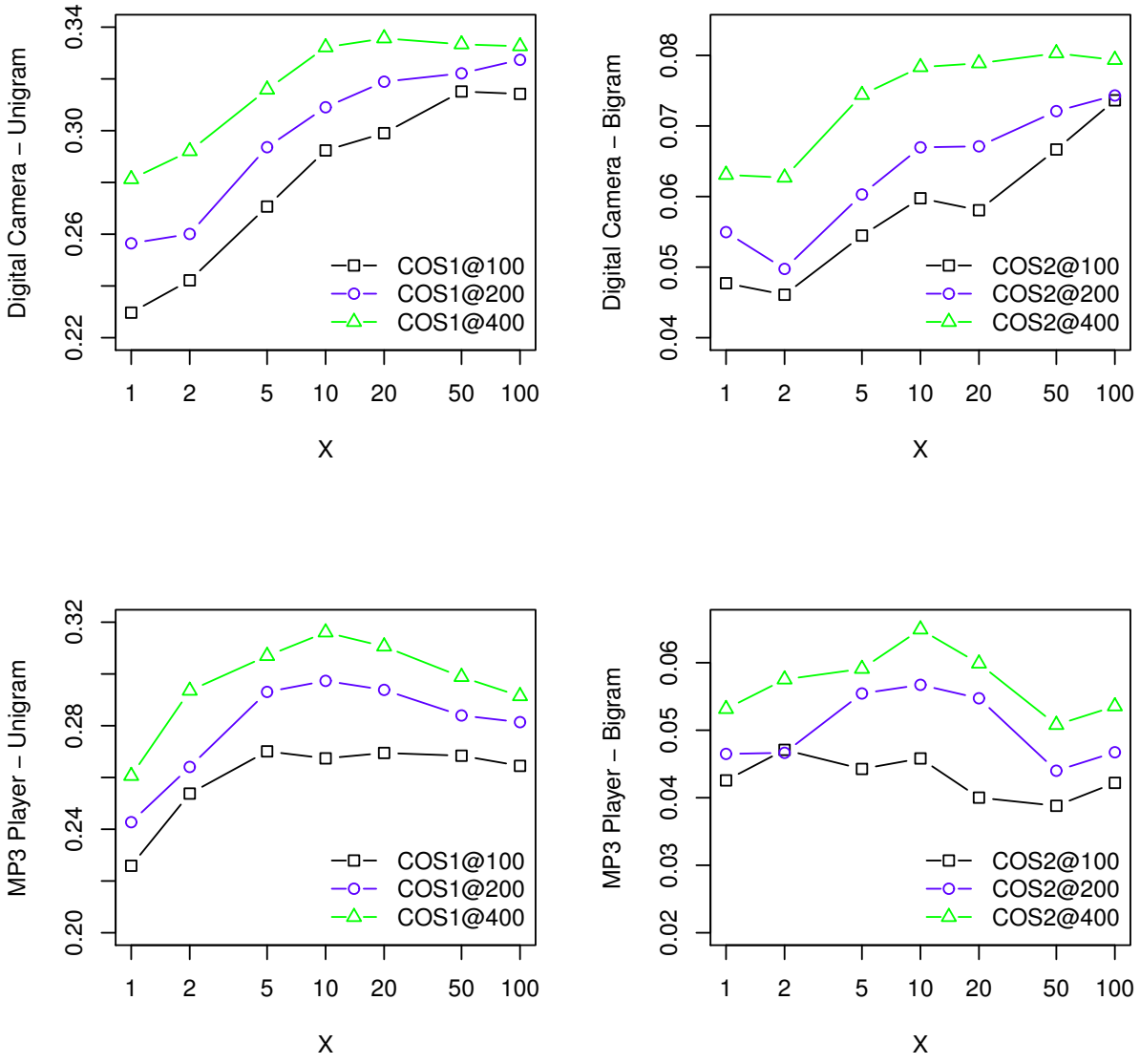


Figure 6.3: TFIDF cosine similarity evaluation results for Translation model with different number (X) of translating products. Upper figures are for digital cameras, and lower figures are for mp3 players. Left figures are results based on unigrams, and right figures are those base on bigrams.

Table 6.7: Unigram and bigram ROUGE @ K for MP3 Players category.

K	Model	ROUGE1-R	ROUGE1-P	ROUGE1-F	ROUGE2-R	ROUGE2-P	ROUGE2-F
	QL	0.278	0.218	0.150	0.0545	0.0308	0.0297
	MEAD	0.202	0.217	0.132	0.0364	0.0242	0.0227
	MEAD-SIM	0.328	0.319	0.196	0.0615	0.0381	0.0367
100	SpecGen	0.303	0.299	0.191	0.0727	0.0480	0.0406
	ReviewSpecGen	0.323	0.320	0.204	0.0650	0.0431	0.0378
	Translation	**0.369**†‡	**0.375**‡	**0.236**†‡	**0.1151**†‡	**0.0742**†‡	**0.0634**†‡
	(increase %)	(+11%)	(+12%)	(+20%)	(+87%)	(+95%)	(+73%)
	QL	0.384	0.213	0.166	0.0812	0.0264	0.0300
	MEAD	0.266	0.171	0.127	0.0453	0.0186	0.0203
	MEAD-SIM	0.434	0.266	0.201	0.0834	0.0290	0.0333
200	SpecGen	0.413	0.273	0.204	0.0913	0.0423	0.0395
	ReviewSpecGen	0.411	0.267	0.197	0.0848	0.0324	0.0344
	Translation	**0.481**†‡	**0.318**†‡	**0.239**†‡	**0.1582**†‡	**0.0664**†‡	**0.0657**†‡
	(increase %)	(+11%)	(+20%)	(+19%)	(+90%)	(+129%)	(+97%)
	QL	0.501	0.186	0.175	0.1159	0.0205	0.0265
	MEAD	0.370	0.154	0.141	0.0517	0.0111	0.0146
	MEAD-SIM	0.560	0.224	0.210	0.1260	0.0207	0.0268
400	SpecGen	0.535	0.221	0.207	0.1228	0.0293	0.0342
	ReviewSpecGen	0.546	0.221	0.204	0.1171	0.0254	0.0314
	Translation	**0.595**†‡	**0.240**†‡	**0.225**†‡	**0.2112**†‡	**0.0431**†‡	**0.0542**†‡
	(increase %)	(+6%)	(+7%)	(+7%)	(+68%)	(+108%)	(+102%)

is different from the TFIDF cosine similarity results; this means that the sentences retrieved by SpecGen are more similar to actual reviews than those retrieved by ReviewSpecGen, but ReviewSpecGen retrieved more "important" relevant words. Translation model outperforms all other models in all measures (18/18), and the increase from MEAD-SIM and ReviewSpecGen is statistically significant in most measures (17/18 and 18/18, respectively). Similar to TFIDF cosine similarity results, the performance difference in bigram measures is clearer than in unigram measures, which means Translation model retrieves bigger fragments of actual reviews well. The increase in unigram ROUGE measures is not as big as that in unigram TFIDF cosine similarity measures, which means that the number of relevant words from Translation model is not very different from other models, but Translation model retrieves much more important relevant words.

We also evaluated retrieved sentences for digital cameras with ROUGE metrics. In general, Translation model outperforms other models in all measures. More specifically, it significantly outperforms MEAD-SIM and ReviewSpecGen in most measures (16/18 and 18/18, respectively). We do not list ROUGE evaluation results for digital cameras since the other patterns are similar to those for mp3 players.

Overall, ROUGE evaluation results are similar to cosine similarity evaluation results in general. The difference between the two metrics is that the TFIDF cosine similarity metric differentiates various models more clearly since they consider importance of words while the ROUGE metric does not; TFIDF cosine similarity metric prefers retrieved text that contains more important words, which is a desired property in such evaluation. On the other hand, ROUGE metric considers various evaluation aspects such as recall, precision, and F1 score, which can possibly help us analyze evaluation results in depth.

In order to reduce computation complexity of Translation model, we proposed to exploit X most promising products that are similar to the query product, instead of all products, under the assumption that similar products are likely to have similar reviews. We performed experiments with different X values to find how many translating products are needed to obtain reasonably good performance. The results are evaluated with TFIDF cosine similarity @ K for unigrams and bigrams, and the results are shown in Figure 6.3. Surprisingly, only a few translating products (e.g., ten) are enough to perform reasonably well especially for mp3 players. These results mean that only a few "good" translating products are enough to translate a candidate sentence well, and the "good" translating products may be selected by their similarity to the query product.

6.9 Conclusion and Future Work

In this work, we studied the problem of automatic relevant review text retrieval for products having no reviews. Relevant review sentences for new or unpopular products can be very useful for consumers who seek for relevant opinions, but no previous work has addressed this novel problem. We proposed several methods to solve this problem, including summarization-based methods such as MEAD and MEAD-SIM and probabilistic retrieval methods such as Specifications Generation model, Review and Specifications Generation model, and Translation model. To evaluate relevance of retrieved opinion sentences in the situation where human-labeled judgments are not available, we measured the proximity between the retrieved text and the actual reviews of a query product. Experiment results show that our novel Translation model indeed retrieves useful sentences and significantly outperforms the baseline methods.

There are limitations in this work. We proposed an automatic evaluation method for the retrieved opinion sentence because manual evaluation is too expensive. However, the automatic evaluation methods do not replace manual evaluation methods since the evaluation requires understanding of sophisticated sentences in user's perspective. In real e-commerce sites, new products sometimes contain completely new features and feature values. However, our work cannot reliably predict opinions for new features or new feature values that hardly appeared in the data. Also, time when reviews were written are not considered in our proposed methods, but people's standard changes as time goes by. For example, "5 megapixels" of image resolution may be very nice 15 years ago, but it is not satisfying at this moment. We may need to consider time to more accurately predict opinions for new products.

In this work, we used a predefined similarity function to compute similarity between products. Our similarity function basically assigns the same weight to each feature in specifications, which is not ideal. Instead of using the predefined one, we can also learn similarity metric between products. For example, we can apply a metric learning technique in [106] to learn similarity metric for products. To generate the training data, we can manually specify which products are actually similar. However, such manual labeling effort may be too expensive as we need to consider multiple criteria across numerous products when we find similar products. Instead, we can make an assumption, *e.g.*, if user reviews are similar then products are similar, where the review similarity can be measured with standard text similarity metrics such as TF-IDF cosine similarity or BM25. Then, we use user reviews of products to learn product similarity in terms of specifications as well as other product data such as product descriptions. In this way, we can automatically learn which aspects of products should be considered more or less important when finding similar products. We leave learning such similarity metric between products as our future work.

Our work opens up a new direction in text data mining and opinion analysis. The new problem of review

text retrieval for new products can be studied from multiple perspectives. First, it can be regarded as a summarization problem as the retrieved sentences need to be central across different reviews. Second, as done in this work, it can also be regarded as a special retrieval problem with the goal of retrieving relevant opinions with product specifications as a query. Finally, it can also be studied from the perspective of collaborative filtering where we would leverage related products to recommend relevant "opinions" to new products, which is a cold-start problem. All these are interesting future directions that can potentially lead to even more accurate and more useful algorithms.

Chapter 7

Conclusion

7.1 Summary

E-commerce has gradually changed the way of buying products in conjunction with the development of Internet. However, user experience in current e-commerce systems is still far from the optimum. This dissertation systematically studies to enhance user experience in e-commerce with joint analysis of user-generated content and product information. In order to assist users in discovering products, (1) we first proposed to leverage user reviews to improve product search accuracy, and then, (2) we proposed to recommend products via inference of implicit intent in social media text. In order to assist users in making purchase decisions, (3) we jointly modeled user reviews and product specifications to augment product specifications with useful knowledge, and (4) we retrieved opinion sentences for new products that do not have any reviews, through joint analysis of user reviews and product specifications.

To improve accuracy for product retrieval, we jointly modeled user reviews and product descriptions since user reviews often bridge vocabulary gap between a user query and product descriptions. The joint model effectively bridged the vocabulary gap while it excluded noise in user reviews. Experiment results indicated that the proposed approach significantly outperformed the state-of-the-art methods. This work serves as the first systematic study of mobile app retrieval task.

To recommend products for social media text that contains implicit intent, we leveraged social media text data to infer user intention. We first built parallel corpora that help us translate implicit intention text into explicit intention text. Then, we inferred user intention in "user status text" using the parallel corpora, and we retrieved products (mobile apps) that satisfy the inferred user intention. The experiment results indicated that the proposed approach was effective and it outperformed the state-of-the-art methods. This work serves as the first study of app recommendation based on social media text with implicit intention.

We jointly modeled user reviews and product specifications to generate augmented specifications so that users can easily understand product specifications. We proposed a novel model that mines useful information about product features from user reviews. The experiment results indicated that the model

effectively retrieved opinion sentences for each product feature, inferred each feature's importance, and extracted special words for each product compared with other products.

To mine opinions for new products that do not have any reviews, we jointly analyzed user reviews and product specifications. We proposed novel probabilistic models that leverage product specifications to estimate similarity between the new product and candidate products. Then, the estimated similarity was used to retrieve relevant opinions for the new product so that consumers can find useful discussions in the opinions. The experiment results indicated that the proposed models indeed retrieved useful sentences and significantly outperformed the baseline methods. This work serves as the first study of opinion retrieval for new products.

Throughout this dissertation, we employed language model-based information retrieval techniques [113] to mine useful knowledge in general. Language models have a property of representing text data in probability distributions, and this property makes the language models versatile so that many probabilistic approaches can be applied to the solutions. In this dissertation, we focused on probabilistic topic models to more effectively mine useful knowledge.

In section 1.2, we discussed the challenges in joint analysis of user-generated content and product information, such as vocabulary gap between two different types of text data, joint analysis with structured product information, and noise in user-generated content. In general, we were able to overcome such challenges by employing topic modeling techniques throughout this dissertation. Different people have their own preferred vocabulary sets. In addition, ordinary people often use informal language in user reviews or social media while product manufacturers often use formal language in order to establish trust in their products. Topic models are able to capture word semantics in text corpora so that they can effectively bridge vocabulary gap between different text data. By carefully designing topic models, we were also able to filter out noise in user-generated content while we could take advantage of similarity between two different text data to model topics. The technique we proposed is also generally applicable to other applications. For example, it can analyze differences and similarities between two different text data with small adjustments. It is applied to products in this dissertation, but it can also be directly applied to other entities such as politicians so that we can align facts about the politicians with opinions about them.

We also applied topic modeling techniques to jointly analyze user-generated content and structured product information. In order to model topics in unstructured text data and structured text data and to align the topics between them, we extended the idea of semi-supervised topic models. We imposed prior knowledge from product specifications to topics as prior probability distribution, and then we expanded the topic words through user reviews to gain users' useful knowledge about product features and the feature

values. Through a hierarchy between features and their values, we could effectively mine useful opinions. We tested our idea in the domain of e-commerce, but it can also be directly or indirectly applied to other domains. The technique is able to jointly analyze unstructured text data and structured text data, where the structured text data consist of key-value pairs.

7.2 Future Directions

There can be many possible future directions for this dissertation. Various future directions for each line of work are already discussed in the corresponding chapter, and we further discuss common future directions here. Throughout this dissertation, we studied joint analysis of user-generated content and product information. While such joint analysis was shown to be quite effective, there may be other kinds of data that also need to be analyzed in conjunction with user-generated content and product information. For example, when we analyze user reviews to mine useful knowledge from them, we may need to consider the time when the user-generated content was generated because people's opinions may change as time goes by. With such temporal analysis, we can more accurately analyze not only the past data but also the future data. As discussed in section 6.9, "5 megapixels" of image resolutions was satisfying 15 years ago, but not anymore. Considering time in joint analysis would enable us to mine more accurate and insightful knowledge from data.

In this dissertation, we performed evaluation based on relevance. That is, when we evaluate the retrieved texts/entities, we measured the accuracy of the retrieved texts/entities based on their relevance to the actual relevant texts/entities. While such evaluation measures how relevant the retrieved information is to the actual relevant information, they do not consider the retrieved information's utility in the user's perspective. For example, users consider multiple criteria such as relevance, value, and brand when they search for products. Ranking products based on those criteria would be more useful to users. To develop such systems, for example, we can borrow concepts such as utility and surplus from economics as in [54]. Through optimization of utility, we can learn what aspects of products users value more so that we can rank products based on the learned criteria. Such idea can also be applied to our work with additional product data such as price and market share. Price information is usually available, and we can sort products by "best selling" in e-commerce sites to estimate market share of them, for example. Then, we can automatically learn what features in specifications and what other criteria are more important in the user's perspective. Search results from such system will help users find more valuable products to them.

Our work that assists users in discovering products can further be studied for virtual assistants. We

studied product search and recommendation given a query, which is made by a user. Interestingly, users make queries to virtual assistants as if they chat with robots, so the queries are likely to be informal. The vocabulary used in such informal queries overlap much with user-generated content such as user reviews and social media text data. Hence, analysis of queries in virtual assistants may benefit from joint analysis with user-generated content, especially when there does not exist enough training data due to the cold-start problem.

We leave all these interesting problems as future work. We created multiple test collections and made them publicly available so that further related studies are enabled.

References

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. Association for Computational Linguistics, 2011.

[2] Azin Ashkan, Charles LA Clarke, Eugene Agichtein, and Qi Guo. Classifying and characterizing query intent. In *Advances in Information Retrieval*, pages 578–586. Springer, 2009.

[3] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 588–596. Springer, 2005.

[4] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 285–294. ACM, 2015.

[5] Lisa Ballesteros and W Bruce Croft. Phrasal translation and query expansion techniques for cross-language information retrieval. In *ACM SIGIR Forum*, volume 31, pages 84–91. ACM, 1997.

[6] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proceedings of ACM SIGIR 1999*, pages 222–229, 1999.

[7] Indrajit Bhattacharya, Shantanu Godbole, and Sachindra Joshi. Structured entity identification and document categorization: two tasks with one joint model. In *Proceedings of ACM KDD 2008*, pages 25–33, 2008.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[9] Andrei Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[10] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[11] Chris Buckley and Ellen M Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

[12] Jay Budzik and Kristian Hammond. Watson: Anticipating and contextualizing information needs. In *Proceedings of the Annual Meeting-American Society for Information Science*, volume 36, pages 727–740. Citeseer, 1999.

[13] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[14] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

[15] Surajit Chaudhuri, Gautam Das, Vagelis Hristidis, and Gerhard Weikum. Probabilistic ranking of database query results. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 888–899. VLDB Endowment, 2004.

[16] Honghua Kathy Dai, Lingzhi Zhao, Zaiqing Nie, Ji-Rong Wen, Lee Wang, and Ying Li. Detecting online commercial intention (oci). In *Proceedings of the 15th international conference on World Wide Web*, pages 829–837. ACM, 2006.

[17] Anindya Datta, Kaushik Dutta, Sangar Kajanan, and Nargin Pervin. Mobilewalla: A mobile application search engine. In *Mobile Computing, Applications, and Services*, pages 172–187. Springer, 2012.

[18] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003.

[19] Arjen P De Vries, Anne-Marie Vercoustre, James A Thom, Nick Craswell, and Mounia Lalmas. Overview of the inex 2007 entity ranking track. In *Focused Access to XML Documents*, pages 245–251. Springer, 2008.

[20] Chrysanthos Dellarocas, Xiaoquan Michael Zhang, and Neveen F Awad. Exploring the value of online product reviews in forecasting sales: The case of motion pictures. *Journal of Interactive marketing*, 21(4):23–45, 2007.

[21] Xiao Ding, Ting Liu, Junwen Duan, and Jian-Yun Nie. Mining user consumption intention from social media using domain adaptive convolutional neural network. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[22] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. Supporting keyword search in product database: A probabilistic approach. *Proc. VLDB Endow.*, 6(14):1786–1797, September 2013.

[23] Junwen Duan, Xiao Ding, and Ting Liu. Mining intention-related products on online q&a community. In *Social Media Processing*, pages 13–24. Springer, 2014.

[24] Wenjing Duan, Bin Gu, and Andrew B Whinston. The dynamics of online word-of-mouth and product salesâĂŤan empirical investigation of the movie industry. *Journal of retailing*, 84(2):233–242, 2008.

[25] Hui Fang, Tao Tao, and ChengXiang Zhai. A formal study of information retrieval heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56. ACM, 2004.

[26] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[27] Kavita Ganesan and ChengXiang Zhai. Findilike: preference driven entity search. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 345–348. ACM, 2012.

[28] Kavita Ganesan and ChengXiang Zhai. Opinion-based entity ranking. *Information retrieval*, 15(2):116–150, 2012.

[29] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, volume 9, pages 1–6. Citeseer, 2009.

[30] Qi Guo and Eugene Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137. ACM, 2010.

[31] Gerald Häubl and Valerie Trifts. Consumer decision making in online shopping environments: The effects of interactive decision aids. *Marketing science*, 19(1):4–21, 2000.

[32] Thomas Hofmann. 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, 1999.

[33] Bernd Hollerit, Mark Kröll, and Markus Strohmaier. Towards linking buyers and sellers: detecting commercial intent on twitter. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 629–632. International World Wide Web Conferences Steering Committee, 2013.

[34] Liangjie Hong and Brian D Davison. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, pages 80–88. ACM, 2010.

[35] Eduard Hovy and Chin-Yew Lin. Automated text summarization in SUMMARIST. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*. MIT Press, 1999.

[36] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of KDD '04*, pages 168–177, 2004.

[37] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760, 2004.

[38] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[39] Bernard J Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: A study of user queries on the web. In *ACM SIGIR Forum*, volume 32, pages 5–17. ACM, 1998.

[40] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[41] Frederick Jelinek. Interpolated estimation of markov source parameters from sparse data. In *Proc. Workshop Pattern Recognition in Practice*, pages 381–397, 1980.

[42] Jaap Kamps, Maarten Marx, Maarten De Rijke, and Börkur Sigurbjörnsson. Xml retrieval: What to retrieve? In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 409–410. ACM, 2003.

[43] In-Ho Kang and GilChang Kim. Query type classification for web document retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 64–71. ACM, 2003.

[44] Max Kaufmann and Jugal Kalita. Syntactic normalization of twitter messages. In *International conference on natural language processing, Kharagpur, India*, 2010.

[45] Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. Comprehensive review of opinion summarization. *Computer Science Research and Tech Reports, University of Illinois at Urbana-Champaign*, 2011.

[46] Hyun Duk Kim and ChengXiang Zhai. Generating comparative summaries of contradictory opinions in text. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 385–394, New York, NY, USA, 2009. ACM.

[47] Reiner Kraft and Jason Zien. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web*, pages 666–674. ACM, 2004.

[48] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of ACM SIGIR '95*, pages 68–73, 1995.

[49] M Lalmas. Xml retrieval (synthesis lectures on information concepts, retrieval, and services). *Morgan and Claypool, San Rafael, CA*, 2009.

[50] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[51] Victor Lavrenko, Martin Choquette, and W Bruce Croft. Cross-lingual relevance models. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 175–182. ACM, 2002.

[52] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.

[53] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 391–400. ACM, 2005.

[54] Beibei Li, Anindya Ghose, and Panagiotis G Ipeirotis. Towards a theory model for product search. In *Proceedings of the 20th international conference on World wide web*, pages 327–336. ACM, 2011.

[55] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.

[56] Henry Lieberman et al. Letizia: An agent that assists web browsing. *IJCAI (1)*, 1995:924–929, 1995.

[57] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, 2004.

[58] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 283–292. ACM, 2013.

[59] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[60] Bing Liu. Sentiment analysis and subjectivity. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010. ISBN 978-1420085921.

[61] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of WWW '05*, pages 342–351, 2005.

[62] Ziyang Liu, Jeffrey Walker, and Yi Chen. Xseek: a semantic xml search engine using keywords. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1330–1333. VLDB Endowment, 2007.

[63] Yue Lu and Chengxiang Zhai. Opinion integration through semi-supervised topic modeling. In *Proceedings of the 17th international conference on World Wide Web*, pages 121–130. ACM, 2008.

[64] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of WWW '09*, pages 131–140, 2009.

[65] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[66] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[67] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[68] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM, 2007.

[69] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of WWW '07*, pages 171–180, 2007.

[70] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. Bad news travel fast: A content-based analysis of interestingness on twitter. In *Proceedings of the 3rd International Web Science Conference*, page 8. ACM, 2011.

[71] Douglas W Oard. A comparative study of query and document translation for cross-language information retrieval. In *Machine Translation and the Information Soup*, pages 472–483. Springer, 1998.

[72] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[73] Paul Ogilvie and Jamie Callan. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 143–150. ACM, 2003.

[74] C. D. Paice. Constructing literature abstracts by computer: techniques and prospects. *Inf. Process. Manage.*, 26(1):171–186, 1990.

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

[76] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, 2008.

[77] Dae Hoon Park, Hyun Duk Kim, ChengXiang Zhai, and Lifan Guo. Retrieval of relevant opinion sentences for new products. In *Proceedings of the 38th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '15. ACM, 2015.

[78] Dae Hoon Park, Mengwen Liu, ChengXiang Zhai, and Haohong Wang. Leveraging user reviews to improve accuracy for mobile app retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '15. ACM, 2015.

[79] Dae Hoon Park, ChengXiang Zhai, and Lifan Guo. Speclda: Modeling product reviews and specifications to generate augmented specifications. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 2015.

[80] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[81] Jovan Pehcevski, Anne-Marie Vercoustre, and James A Thom. Exploiting locality of wikipedia links in entity ranking. In *Advances in Information Retrieval*, pages 258–269. Springer, 2008.

[82] Isidro Peñalver-Martínez, Rafael Valencia-García, and Francisco García-Sánchez. Ontology-guided approach to feature-based opinion mining. In *Natural Language Processing and Information Systems*, pages 193–200. Springer, 2011.

[83] Joaquín Pérez-Iglesias, José R Pérez-Agüera, Víctor Fresno, and Yuval Z Feinstein. Integrating the probabilistic models bm25/bm25f into lucene. *arXiv preprint arXiv:0911.5046*, 2009.

[84] Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

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

[86] Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 21–30. Association for Computational Linguistics, 2000.

[87] Stephen E Robertson. The probability ranking principle in ir. *Readings in information retrieval*, pages 281–286, 1997.

[88] Andrew J Rohm and Vanitha Swaminathan. A typology of online shoppers based on shopping motivations. *Journal of business research*, 57(7):748–757, 2004.

[89] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[90] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[91] Fei Song and W Bruce Croft. A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321. ACM, 1999.

[92] Weifeng Su, Jiying Wang, Qiong Huang, and Fred Lochovsky. Query result ranking over e-commerce web databases. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 575–584. ACM, 2006.

[93] Vanitha Swaminathan, Elzbieta Lepkowska-White, and Bharat P Rao. Browsers or buyers in cyberspace? an investigation of factors influencing electronic exchange. *Journal of Computer-Mediated Communication*, 5(2):0–0, 1999.

[94] Bin Tan, Atulya Velivelli, Hui Fang, and ChengXiang Zhai. Term feedback for information retrieval with language models. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 263–270. ACM, 2007.

[95] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169. ACM, 2006.

[96] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM, 2008.

[97] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW '08*, pages 111–120, 2008.

[98] Anne-Marie Vercoustre, James A Thom, and Jovan Pehcevski. Entity ranking in wikipedia. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1101–1106. ACM, 2008.

[99] Hanna M Wallach, David Minmo, and Andrew McCallum. Rethinking lda: Why priors matter. 2009.

[100] Norman Walsh, Mary Fernández, Ashok Malhotra, Marton Nagy, and Jonathan Marsh. Xquery 1.0 and xpath 2.0 data model (xdm). *W3C recommendation, W3C (January 2007)*, 2007.

[101] Hongning Wang, Yue Lu, and Chengxiang Zhai. 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, 2010.

[102] Jinpeng Wang, Gao Cong, Xin Wayne Zhao, and Xiaoming Li. Mining user intents in twitter: A semi-supervised approach to inferring intent categories for tweets. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[103] Tao Wang, Yi Cai, Guangyi Zhang, Yu Liu, Junting Chen, and Huaqing Min. Product feature summarization by incorporating domain information. In *Database Systems for Advanced Applications*. Springer, 2013.

[104] David Watson and Auke Tellegen. Toward a consensual structure of mood. *Psychological bulletin*, 98(2):219, 1985.

[105] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM, 2006.

[106] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2003.

[107] Huahai Yang and Yunyao Li. Identifying user needs from social media. Technical report, IBM Tech Report. goo. gl/2XB7NY, 2013.

[108] Qiang Ye, Rob Law, and Bin Gu. The impact of online user reviews on hotel room sales. *International Journal of Hospitality Management*, 28(1):180–182, 2009.

[109] Xing Yi and James Allan. A comparative study of utilizing topic models for information retrieval. In *Advances in Information Retrieval*, pages 29–41. Springer, 2009.

[110] Emine Yilmaz and Javed A Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111. ACM, 2006.

[111] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. App recommendation: a contest between satisfaction and temptation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 395–404. ACM, 2013.

[112] Jianxing Yu, Zheng-Jun Zha, Meng Wang, Kai Wang, and Tat-Seng Chua. Domain-assisted product aspect hierarchy generation: towards hierarchical organization of unstructured consumer reviews. In *Proceedings of EMNLP 2011*, pages 140–150, 2011.

[113] ChengXiang Zhai. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141, 2008.

[114] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.

[115] ChengXiang Zhai, Atulya Velivelli, and Bei Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 743–748. ACM, 2004.

[116] Xue Zhang, Hauke Fuehres, and Peter A Gloor. Predicting stock market indicators through twitter "i hope it is not as bad as i fear". *Procedia-Social and Behavioral Sciences*, 26:55–62, 2011.

[117] Lina Zhou and Pimwadee Chaovalit. Ontology-supported polarity mining. *Journal of the American Society for Information Science and technology*, 59(1):98–110, 2008.

[118] Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen. Mobile app recommendations with security and privacy awareness. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 951–960. ACM, 2014.