Mining User Reviews for Mobile App Comparisons

YUANCHUN LI, BAOXIONG JIA, YAO GUO*, and XIANGQUN CHEN, Peking University, China

As the number of mobile apps keeps increasing, users often need to compare many apps, in order to choose one that best fits their needs. Fortunately, as there are so many users sharing an app market, it is likely that some other users with the same preferences have already made the comparisons and shared their opinions. For example, a user may state that an app is better in power consumption than another app in a review, then the review would help other users who care about battery life while choosing apps. This paper presents a method to identify comparative reviews for mobile apps from an app market, which can be used to provide fine-grained app comparisons based on different topics. According to experiments on 5 million reviews from Google Play and manual assessments on 900 reviews, our method is able to identify opinions accurately and provide meaningful comparisons between apps, which could in turn help users find desired apps based on their preferences.

CCS Concepts: • **Applied computing** → *Document analysis*; • **Information systems** → *Retrieval models and ranking*; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools*;

Additional Key Words and Phrases: Mobile application; user review; comparative opinion; text processing

ACM Reference Format:
Yuanchun Li, Baoxiong Jia, Yao Guo, and Xiangqun Chen. 2017. Mining User Reviews for Mobile App Comparisons. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 75 (September 2017), 15 pages.
DOI: http://doi.org/10.1145/3130935

1 INTRODUCTION

With the prevalence of mobile devices such as smartphones, mobile applications (*apps* in short) have seen widespread adoption, with over two million apps available in both Google Play and Apple App Store, while billions of downloads have been accumulated [24, 25]. Many of these apps are similar in functionality, thus users often need to choose one that best fits their needs.

Major app markets such as Apple App Store and Google Play usually rank or recommend apps based on the relevance of an app, its popularity (such as the number of total installations) and user ratings (such as average scores given by all users). However, these kinds of information may not be enough to empower a user to download and install their desired apps in the first attempt. Thus users often have to install multiple apps and try them out for a while in order to make comparisons, before they find the right one.

Many recent studies have attempted to provide more fine-grained app comparisons while taking into consideration of properties such as privacy [4] and power consumption [9]. For example, PrivacyGrade [4] ranks apps based on their privacy friendliness. EcoDroid [9] presented a method to rank Android apps according to their

*Corresponding author.

Authors' affiliation and address: Y. Li, B. Jia, Y. Guo and X. Chen, Key Laboratory on High-Confidence Software Technologies (Ministry of Education), School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China.

2474-9567/2017/9-ART75 $15.00
DOI: http://doi.org/10.1145/3130935

power consumption. In addition, some third-party websites [1, 28] contain manually-written articles comparing apps, but the number of reviewed apps is typically limited and some of the comparisons may also be biased.

Providing fine-grained app comparisons is challenging because it is sometimes difficult to understand these fine-grained characteristics (such as privacy and energy) for each app. It is also normally impossible to evaluate each app from every possible aspect, as many aspects that users would like to compare cannot be quantified.

The main idea of this paper is that *we take advantage of publicly-available user reviews as the source for app comparisons*. Many popular apps have been reviewed by thousands of users in major app markets, which can be publicly accessed. Users often express their preferences toward an app in their reviews, in many cases even comparing this app with other apps (i.e., *comparative reviews*).

With comparative reviews that compare two apps directly, it can reveal more detailed comparisons between two apps, which explains why other users like/dislike a particular app. For example, a user may state that an app is better in power consumption or quicker in response time than another app in the review, which provides more detailed opinions on these apps. As a result, these comparative reviews can be very helpful in summarizing app comparisons.

This paper proposes a method to identify and summarize comparative opinions from large-scale user reviews automatically. A *comparative opinion* is defined as a comparative sentence within a user review, which compares the current app with another app, while expressing the reviewer's sentiment on a chosen topic such as battery or performance. In order to accurately identify comparative opinions from a large number of user reviews, we introduce two techniques: *comparative review identification based on app alias matching and comparative pattern matching*, and *opinion summation based on topic analysis and preference analysis*.

The first task is to identify sentences that compare two apps directly from a large number of user reviews. We introduce a comparative review identification method based on app alias matching and comparative pattern matching. Users often use abbreviated names (i.e. aliases) instead of the full name when mentioning an app, for example *fb* for *Facebook* and *insta* for *Instagram*. Based on the insight that the aliases of an app appear frequently in the app's own reviews while less frequently in other apps' reviews, we introduce a model named *tf-relevance graph* to find the aliases. We also make use of sequential pattern mining to detect the common patterns of comparative sentences. With the aliases and comparative patterns, we are able to identify comparative reviews from a large number of reviews.

We then summarize the opinions from the comparative reviews based on the topics and preferences expressed by the reviewers. We use a keyword-based approach to conduct topic analysis, which does not require manually labeling reviews and supports user-defined arbitrary topics. For each identified topic, we extend an existing opinion mining algorithm for comparative sentences to recognize the preferred apps in the compared topics expressed by the reviewer.

We evaluate our method on about 5 million user reviews for popular apps from 6 categories collected from Google Play. Because there is no ground truth available, we recruited nine volunteers to evaluate the accuracy of our detection methods, with each piece of results examined by three volunteers. (The majority decision is used as the ground truth when they have different opinions.) We have identified over 10,000 comparative reviews on these apps, with an accuracy of 93.7%. We then identified more than 4,000 different topics within these comparative reviews, with an accuracy of about 80%. For each correctly identified topic from the reviews, we are able to determine the user preference with an accuracy of 82.8%.

This paper makes the following research contributions:

- We introduce an automated method to identify comparative sentences from large-scale user reviews. With a list of app aliases, we are able to identify user reviews directly comparing two apps with a high accuracy.
- We summarize comparative opinions based on comparative reviews into common topics including battery, UI, etc. With these topics, we can provide fine-grained comparisons on two apps from different perspectives.

- We tested our method with an experiment on about 5 million user reviews collected from Google Play. Based on an evaluation on 900 reviews with manually labeled ground truth, our method can identify comparative reviews accurately and extract comparative opinions with an acceptable precision.

2 RELATED WORK

2.1 Mobile App Comparison

App comparison has attracted the interests of industry and academia due to its benefits for app recommendation. Both Google Play [25] and App Store [24] provide rankings of apps based on popularity or user rating, which are general comparisons between apps.

Some research approaches have considered detailed aspects of mobile apps in comparison. For example, PrivacyGrade [4] gives a privacy score to each Android app by measuring the gap between the app's actual behavior and users' expectation. EcoDroid [9] is able to rank Android apps according to their power consumption. However, these approaches only consider a small subset of the aspects concerned by users and need special treatment to evaluate each aspect. In contrast, our work is able to include various aspects, as long as they are discussed in user reviews.

2.2 Mobile App User Review Analysis

Several studies have examined mobile app user reviews for different purposes. Some approaches [16, 22] are focused on discovering the patterns or conventions of users writing reviews. Some of them were interested in understanding users' attitude, for example, what users are complaining about [8, 12] and what features in an app are liked by users [6]. There are also some approaches to facilitate developers or analysts to read the reviews, such as AR-miner [3] which can help find informative reviews and MARK [23] which can help search and view related reviews.

Our work is also focused on analyzing mobile app user reviews, however for a different purpose in fine-grained app comparison.

2.3 Opinion Mining

Opinion mining techniques have been well-studied in natural language processing. The comparative sentence identification problem was first studied by Jindal *et al.* [11] in 2006. They presented a novel integrated pattern discovery and supervised learning approach to identifying comparative sentences from text documents. Based on this work, Ganapathibhotla and Liu [5] proposed a method to extract comparative opinions. Park *et al.* [18] used machine learning to identify comparative claims in scientific articles based on semantic and syntactic features. Li *et al.* [14] worked on comparative question identification and comparable entity extraction from online user questions. ReviewCollage [10] summarized review comments posted in online websites to highlight the similarities and differences between the entities. Xu *et al.*[27] dealt with a similar problem to ours, which is extracting customers' comparative opinions between mobile phones from Amazon user reviews. However they only processed 1,347 selected customer reviews from 33 types of mobile phones, while they did not focus on identifying comparative reviews from large scale data.

Sentiment analysis is another useful technique to analyze user's attitude in a sentence. Pang *et al.*[17] summarized many existing techniques in 2008. In recent years, researchers also tried to use deep learning to improve sentiment analysis [21].

Analyzing the topics of the text document is also a popular research direction. LSA [13], PLSA [7] and LDA [2] can be used to train a topic model and predict the topics of text based on the model. Particularly, AppLDA proposed by Park *et al.* [19] is a topic model tailored for app descriptions and user reviews. Aside from using a topic model, some keyword-based methods can be used for analyzing the correlation between a text document

and a given topic. For example, query expansion [20, 26] is used by many search engines to retrieve texts related to a keyword query.

We extend existing opinion analysis techniques to address the problem of comparative opinion mining in mobile app user reviews.

3 APPROACH OVERVIEW

3.1 Comparative Opinions

One key concept we introduce in this paper is *comparative opinion*, which represents the user opinions on an app in comparison to other apps. Comparative opinions can be found within reviews written under each app, where users may compare the app to other apps with similar functionalities.

We first give a definition of comparative opinion:

Definition 1. *A **comparative opinion** in a user review is defined as a tuple* $\{\langle opponent\rangle, \langle topic\rangle, \langle preferred\rangle\}$*, where* $\langle opponent\rangle$ *is the entity (app) that the current app is compared against,* $\langle topic\rangle$ *is an aspect that the review is talking about, and* $\langle preferred\rangle$ *can be either "current" or "opponent", meaning whether the current app or the opponent app is preferred by the user. Accordingly, the user review that carries the comparative opinion is called a **comparative review**.*

For example, one of *Firefox*'s user reviews says "Slower page loading than chrome". This review directly compares *Firefox* with *Chrome*. This review is on the performance of web browsers, and the opponent app *Chrome* is preferred by the user. Thus the opinion in this review can be represented as {*Chrome*, *performance*, *opponent*}, which means that the user prefers *Chrome* because it is better in performance than *Firefox*.

Due to the large scale of publicly-available user reviews in app markets, there are a lot of fine-grained comparative opinions that can be extracted for benign use.

3.2 Goal and Challenges

The goal of our approach is to extract and summarize comparative opinions from large-scale user reviews. We first want to identify comparative sentences from user reviews, which needs to apply some NLP techniques. However, being a comparative sentence does not necessarily make it a comparison between two apps, thus we need specific algorithms to identify comparative reviews involving two apps. We also want to further identify more information from these reviews, including the topic in the comparison and which app is better (opinion mining).

We face the following main challenges:

- *How to identify user reviews that compare two apps?* First of all, comparative sentence identification is still an open question. There are no mature algorithms to determine whether a sentence is making a comparison. When it comes to comparative user reviews, the problem becomes even more challenging, because the reviews are informal and the entities in comparison are often an abbreviation (alias) of app names. We not only need to identify comparative sentences, but also need to make sure they are comparing two apps.
- *How to understand user preferences/opinions based their comparative reviews?* For each review, we need to know which aspects (topics) are compared and which app is better for each aspect in the reviewer's opinion. The aspects compared in a review include both common topics such as battery and UI, as well as specific topics such as the feature of an app.

3.3 Technique Overview

We introduce a two-step approach to extract comparative opinions from large-scale user reviews, as shown in Figure 1. In the first step, we identify comparative reviews from large-scale user reviews by matching the aliases

Fig. 1. The overview of our approach.

of the apps (entities) and the patterns of comparative sentences. In the second step, we conduct topic analysis and preference analysis on the set of identified reviews.

In the comparative review identification step, finding the app-alias pairs and the comparative patterns is the key. We introduce an automated method to identify the aliases of an app based on the correlation between term frequency and app relevance. For comparative patterns, we summarize a set of common patterns from known comparative reviews. The comparative pattern used in our approach is a sequence of PoS (Part-of-Speech) tags tailored for user reviews.

In the opinion summation step, we make use of a keyword-based method to extract the topics included in each review. The keyword-based method is able to deal with not only a set of common topics but also any arbitrary topic defined by users. We then extend an existing opinion mining algorithm to recognize the reviewer's preferred app for each compared topic.

4 IDENTIFYING COMPARATIVE REVIEWS

The first step in our approach is to find all comparative reviews from a large number of user reviews from Google Play. A comparative review must satisfy the following two conditions:

(1) Another app is referenced in the review;
(2) The referenced app is an entity in a comparison.

Because comparative sentence identification is still in research stage, we do not rely solely on existing comparative sentence recognition methods such as in Jindal's work [11]. Instead, we take advantage of a special property of sentences on app comparison, as each such sentence must involve another app. Based on the observation, we can use the existence of the name of another app to identify comparative sentences in a review.

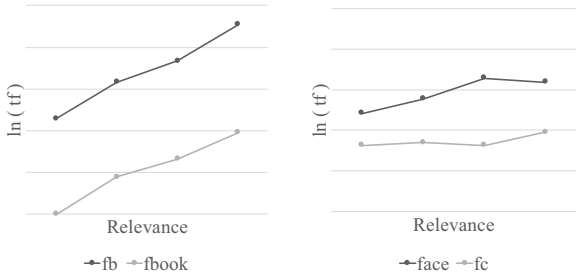
However, app names are not always used as is. Because the original name of an app might be too long or too formal, users usually use a short alias instead. For example, many users prefer to use *fb/FB* to represent *Facebook*. We assume an app is referenced if the alias or the original name of the app appears in the review. Thus we first need to design a method to find the aliases of an app.

To make sure the referenced app is an entity in a comparison, we try to match the review sentence with some comparative patterns. If the sentence matches a comparative pattern and the alias matches the entity in the pattern, the review is identified as a comparative review.

4.1 Alias Identification

An alias of an app is a term used by users to reference the app in their reviews. Usually, an alias is an abbreviation of the app's original name, but it does not follow a common convention. For example, *Instagram*'s often-used alias *insta* is the prefix of the app's name, *fb* is a subsequence (or word initials) of the original name *Facebook*, and *cm*, which is one of *Clean Master*'s aliases, is the acronyms of the original name. Considering all possible aliases in comparative review identification is very time-consuming and it may produce many false positives, thus we need an automated method to find the correct aliases of each app first.

We introduce a method to find the aliases of each app by analyzing the reviews. Our method is based on the insight that the aliases of an app appear more frequently in the app's own reviews while less frequently in other



(a) Correct aliases: *fb* and *fbook*. (b) Incorrect aliases: *face* and *fc*.

Fig. 2. The *tf-relevance graph*s of four aliases of *Facebook*. *Note that the X-axis represents four sets of apps with different relevance levels, which are all apps, apps in the* Social *category, apps similar with Facebook and Facebook itself from left to right. The Y-axis represents the logarithmic value of* tf*, where* tf *is the term frequency of the alias in the apps' reviews.*

Table 1. Examples of aliases for some popular apps.

Package	**Original Name**	**Aliases**
com.facebook.katana	Facebook	fb, fbook, facbook
com.android.chrome	Chrome Browser Google	chrome browser, chrome, crome
com.whatsapp	WhatsApp Messenger	whatsapp, watsapp, whatsap, whatapp
com.instagram.android	Instagram	insta, ig, instgram
com.cleanmaster.android	Clean Master (Boost & AppLock)	cm, clean master, cleanmaster
com.adobe.psmobile	Adobe Photoshop Express	photoshop, adobe photoshop, photoshop express
cn.wps.moffice_eng	WPS Office + PDF	wps, office, wps office
com.outfit7.talkingtom	Talking Tom Cat	tom cat, talking tom, talking cat, tom, taking tom

apps' reviews. More specifically, if a term *alias_A* is an alias of app *A*, given another app *B*, the more relevant between app *A* and app *B*, the more frequently will *alias_A* appears in *B*'s user reviews. For example, *fb* frequently appears in *Facebook*'s reviews, less frequently in *Snapchat* (another social network app)'s reviews, and far less frequently in *Pedometer* (a fitness app)'s reviews.

Tf-idf, short for term frequency-inverse document frequency, is intended to reflect how important a word is to a document in a collection or corpus. In our case, *tf-idf* can reflect how important a word is to an app among all apps. Based on the idea of *tf-idf*, we introduce a model named *tf-relevance graph* to describe the positive correlation between term frequency and the relevance between apps.

We considered four levels of relevance in our approach. Given an app A, the set of all apps is the least relevant to app *A*, then the apps in the same category with *A*, the apps considered as similar to *A* by the market (Google Play), and finally the app *A* itself. The term frequency (*tf*) of term *t* in a set of apps' reviews *R* is calculated as the number of reviews containing *t*, divided by the total number of reviews, i.e.:

$$tf(t, R) = \frac{|\{r \in R \ : \ r \ contains \ t\}|}{|R|}$$

Figure 2 shows several examples of *tf-relevance graph*s of *Facebook*'s aliases, in which we can see a positive relevance between *tf* and *relevance* for correct aliases like *fb* and *fbook*, and vice versa. In contrast, incorrect aliases such as "face" or "fc" do not show positive relevance.

Table 2. Comparative patterns and examples. *Note that* CIN *stands for comparative prepositions,* CV *stands for comparative verbs and* APP *stands for app references.*

Pattern	Matched sequence	Original review
*JJR * CIN * APP*	better than uber	Better than uber, They don't need a GPS to get you home.
*RB JJ * CIN APP*	more stable than yahoo	Much more stable than yahoo messenger.
*CV * CIN APP*	prefer over ola	No wonder why people prefer Uber over Ola.
CV VBG APP	prefer using opera	I prefer using Opera than other browsers.
CV APP	beats youtube	Quality is great and it sure beats youtube, ...
*VB VBN * APP*	be compared facebook	..., it can be compared to Facebook, and it doesn't cut off too much credit.
*APP * VBZ * JJS*	airdroid is the best	..., airdroid is the best multipurpose app out there at the moment.
*APP * VBZ * JJR*	qs is better	Team viewer QS is better option.

Table 1 shows some alias examples for popular apps identified using our method. Beside typical aliases such as "fb" or "cm", we can see that some misspelled words such as "crome", "whatsap" and "taking tom" can also be identified as correct aliases.

After finding the aliases of each app, we search for the aliases in all reviews. If an alias appears in a user review, the app is considered as being referenced in the review.

4.2 Comparative Patterns

The apps referenced in a review are not always compared with the current app, instead it might be referenced only because of some functional relevance. For example, a reviewer may say "I will share this app to my Facebook.", which references *Facebook* while not comparing. We introduce a comparative pattern matching step to determine whether the app referenced in a review is in fact used in comparison.

A *comparative pattern* is defined as a sequence of PoS (Part-of-Speech) tags. For example, *"JJR IN"* represents a pattern in which a comparative adjective followed by a preposition, such as "better than" and "slower than". We extend the list of common PoS (Part-Of-Speech) tags to meet the requirements of comparative review identification. More specifically, we introduce three comparative PoS tags: *CV* (comparative verbs, e.g. prefer, recommend), *CIN* (comparative prepositions, e.g. than, over) and *APP* (app references, including app names and aliases, e.g. facebook, fb).

We examined 300 manually-labeled comparative reviews to summarize comparative patterns. First, we convert each comparative review to a sequence of PoS tags. There are some common subsequences shared among the converted PoS tag sequences. Then we tried to represent the common subsequences with a finite set of patterns. To make the patterns more flexible, we used a wildcard character to represent a list of arbitrary words. Finally, we got 8 patterns that are able to cover most comparative subsequences, which is shown in Table 2.

With the common comparative patterns, we are able to determine whether a review is making a comparison by matching these review with the patterns. The main process involves the following steps:

(1) Preprocessing each review, including replacing non-English characters and stemming;
(2) PoS tagging, by labeling each word in a sentence with the standard PoS tags (JJ, VB, etc.) and our customized comparative PoS tags (CV, APP, etc.). Note that there might be multiple PoS tags for each word.
(3) Matching the PoS tag sequence with comparative patterns. We use depth-first search to find all word sequences in a review that match a comparative pattern.

The reviews that match at least one comparative pattern are identified as comparative reviews. Table 2 shows some examples of identified comparative reviews with the matched patterns.

Table 3. Selected common topics and their corresponding keywords.

Topic	Keywords
Performance	effective, powerful, smooth, fluid, reliable, unresponsive, wait, slow, efficient, speed, fast, quick, responsive, consistent
Battery	juice, battery, drain, hog, consume, overheat, heat, consumption, ruin, downgrade, power
Stability	stable, crash, freeze, hang, break, close, stop, restart, shutdown, bug, tap, fix, lag, flaw
Usability	option, smart, intuitive, instruct, experience, friendly, entertain, service, custom, personal
UI	frame, effect, ui, innovative, creative, design, icon, gui, interaction, button, interface, style
Memory	memory, lightweight, light, weight, heavy, ram, mb, gb, space
Ads/spam	commercial, ads, advertisement, advert, advertise, notification, filter, spam
Price	expensive, cheap, affordable, inexpensive, price, cost, bill, fare
Security	secure, safe, privacy, security

5 SUMMARIZING COMPARATIVE OPINIONS

After identifying comparative reviews, we analyze each review to summarize the opinions expressed by the user. According to our definition, a comparative opinion contains an opponent app, a topic and an expressed preference. As the opponent app has been identified through alias matching, we make use of existing NLP techniques to extract the topic and preference.

5.1 Topic Analysis

The goal of topic analysis is to summarize the topics that the user is talking about in the review. Knowing the topics of each user review enables us to find the most relevant user reviews based on a user's concerned aspects.

We introduce a keyword-based approach to analyze the topics in reviews. Its basic idea is compiling a list of keywords for each topic. Thus whether a document contains a topic depends on the frequency of the corresponding keywords in the document. In our case, the document is a short user review, thus a review is considered to be including a specific topic if there is at least one corresponding keyword appearing in the review.

We first consider a set of common topics usually concerned by users. An existing work [23] has summarized some topics and the corresponding keywords. We extend the mapping by adding more topics and more keywords for each topic. We ended up with a list of 9 common topics as shown in Table 3.

Moreover, we support user-defined arbitrary topics using keyword expansion, which can be used to find related keywords based on a given keyword. It has been used by search engines to expand the search query. We make use of Word2Vec [15] to expand a user-defined topic to a set of keywords. Word2Vec generates an N-dimension vector for every single word in the text data, where the vectors of words having similar meanings stay closer in the N-dimension space. Based on the vector set, we can find a set of related keywords starting from one topic word.

5.2 Preference Analysis

The goal of preference analysis is to infer which app is preferred by the reviewer in a comparative review.

As a review is given to the current app, the sentiment in the review often reflects the user's attitude towards the current app, i.e. the user should express a positive sentiment if he/she prefers the current app and a negative sentiment if she prefers the *opponent* app. One simple and effective way to determine the sentiment in a review is to look at the rating given by the reviewer. Google Play allows users to rate an app from one star to five stars, where a one-star rating represents the most negative sentiment while a five-star rating means the most positive.

For most short user reviews, we can directly use the number of stars to determine whether the reviewer prefers the current app.

However, sometimes a review could be a complex sentence, in which the reviewer expresses multiple opinions on different aspects, while the rating is given to the whole review. For example, one of *Lyft*'s reviews says:

> *Little bit more expensive than uber, but opens much earlier. I live in a college town not specifically listed by uber or lyft, meaning uber doesn't open till 8am. I can usually get a lyft within half an hour of finishing my night shift instead of waiting 2 hours for uber, and the dollar is worth the difference.*

The review is rated five stars, but the sentence *"Little bit more expensive than uber"* expresses a negative attitude towards *Lyft*'s price. In order to deal with these cases, we introduce a sentence-level preference analysis algorithm to extract the fine-grained opinions.

We extended the approach proposed by Ganapathibhotla and Liu [5] to analyze the fine-grained opinions expressed in user reviews. Similar to their work, we categorize comparative relations to "increasing" and "decreasing" (e.g. "more than" is increasing and "less than" is decreasing) and categorize the topic keywords to positive and negative (e.g. "cheap" is positive and "expensive" is negative).

However, one special feature of mobile app reviews is that many sentences are incomplete because users often omit the subject in comparison. Thus we also need to label the missing subject for each comparative pattern. For example, suppose "Better than *B*" and "*B* is better" are two reviews of app *A*, the subject of the first review is *A* (the current app), while the subject of the second review is *B* (the *opponent* app).

The preference towards the current app is calculated as:

$$Pref_{review} = Pref_{comparative} * Pref_{keyword} * Pref_{subject}$$

where:

$$Pref_{comparative} = \begin{cases} 1, & \text{for increasing comparative} \\ 0, & \text{for neutral comparative } a = 1 \\ -1, & \text{for decreasing comparative} \end{cases}$$

$$Pref_{keyword} = \begin{cases} 1, & \text{for positive/neutral keyword} \\ -1, & \text{for negative keyword} \end{cases}$$

$$Pref_{subject} = \begin{cases} 1, & \text{current app as the subject} \\ -1, & \text{the } \textit{opponent} \text{ app as the subject} \end{cases}$$

For example, *A*'s review "More expensive than *B*" shows a negative opinion, because it has an increasing comparative "more than" ($Pref_{comparative}$ = 1) and a positive topic keyword "stable" ($Pref_{keyword}$ = -1), and the current app is the subject ($Pref_{subject}$ = 1).

6 EVALUATION

6.1 Experimental Setups

Our experiment is based on about 5 million reviews crawled from Google Play in November 2015. The reviews belong to 2,311 popular apps (packages) from 6 categories. The apps are selected in descending order based on the number of reviewers, and the reviews are selected in descending order of "helpfulness" (the default metric used by Google Play in sorting reviews).

Because some apps may contain a huge number of reviews (for instance, Facebook has millions of reviews), analyzing all reviews for each app is very time-consuming. In order to control the scale of the experiment data, while maintaining an even distribution on the number of reviews per app, we decided to collect at most 4,000

Table 4. The number of reviews and apps in our dataset.

Category	#Reviews	#Apps
Tools	2,027,217	1,071
Photography	1,006,811	396
Communication	727,347	269
Media&Video	542,257	223
Social	505,057	213
Transportation	115,794	139
Overall	4,924,483	2,311

Table 5. The number of identified comparative reviews and apps involved in comparisons. *Note that the* # Comparative Reviews *column is the number and percentage of identified comparative reviews for each app category, and the* # Compared Apps *column is the number and percentage of apps involved in comparisons for each category.*

Category	# Compatative Reviews	# Compared Apps
Tools	3,064 (1.5 ‰)	599 (55.9%)
Photography	1,507 (1.5 ‰)	260 (65.7%)
Communication	3,387 (4.7 ‰)	211 (78.4%)
Media&Video	1,019 (1.9 ‰)	137 (61.4%)
Social	1,191 (2.4 ‰)	124 (58.2%)
Transportation	331 (2.9 ‰)	39 (28.1%)
overall	10,499 (2.1 ‰)	1,370 (59.3%)

reviews for each app. Table 4 shows the detailed distribution of the reviews in each category. On average, there are 2,130 reviews for each app in our dataset.

6.2 Distribution of Comparative Reviews

We first run a pass of comparative review identification to identify potential comparative reviews.

Table 5 shows the number of identified reviews and the corresponding apps involved in comparisons. Overall, we have identified 10,499 comparative reviews from a total of 4,924,483 reviews, which is about a proportion of 1/500. Considering the fact that many reviews only contain a few words such as “Great app” or “Very useful”, the proportion of comparative reviews is reasonable.

Among the 10,499 identified reviews, 43.2% of them (4,535 reviews) use the alias names instead of the original names. Especially for some apps with a long original name (such as “Chrome Browser Google”), almost all reviews use the aliases (such as “chrome”).

In these comparative review, 1,370 apps are explicitly compared, which covers about 59.3% of the apps we have studied. Given the fact that many popular apps have millions of reviews (we have only crawled at most 4,000 reviews for each app), we can potentially obtain much more comparative reviews through a market-scale analysis.

The proportion of comparative reviews differs from category to category. The *Communication* category has the highest proportion of comparative reviews, with 4.7 ‰ of the reviews identified as comparative and involving 78.4% of the apps. The *Transportation* category has a higher number of comparative reviews and a lower number of involved apps. Particularly, the most comparative reviews are found in taxi booking apps such as *Uber* and *Lyft*.

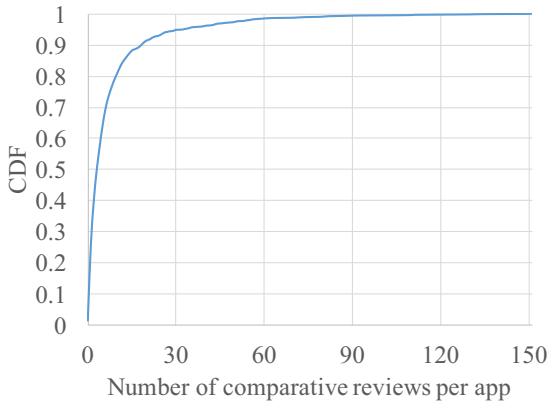


Fig. 3. Distribution (CDF) for the average number of comparative reviews for each app.

The proportion of comparative reviews is consistent with the level of competitive intensity for each category (and each class of apps).

The number of identified comparative reviews differs from app to app. Figure 3 shows the cumulative distribution function (CDF) for the number of identified comparative reviews per app, in which we can see most comparative reviews belongs to a small set of apps. In particular, 20% of the apps have more than 10 comparative reviews and 5% apps have more than 30 comparative reviews. The result is in accord with our intuition because most people are using the popular apps, thus most comparisons are made towards these popular apps.

6.3 Accuracy of Comparative Review Identification

We then evaluate the accuracy of comparative review identification. Because it is difficult to obtain the ground truth of all comparative reviews, we only evaluate the precision in this step.

We selected 5 apps with most comparative reviews in each category, and manually examined 30 user reviews that are identified as comparative for each app. In total, we manually checked 900 reviews for 30 apps.

In order to avoid the bias in manual evaluation, we recruited six volunteers to examine each review. Each review is examined by three volunteers, such that each reviewer labeled exactly 450 reviews. Each volunteer will decide whether a review is actually comparing two apps. For more than 95% of the reviews, the three volunteers gave the same decision. When three volunteers did not agree with each other, we take the majority opinion as the ground truth in the evaluation result.

The precision of comparative review identification is shown in Table 6. Within the 900 reviews examined, 843 of them are correct comparative reviews, showing a precision of 93.7%. The accuracy for the *Tools* category and *Media&Video* is relatively low, because some apps in these categories use very general names such as “task killer”, “wifi analyzer”, “music player” etc. Because these terms are identified as aliases of a specific app while often used to represent a set of such kind of apps, it brings some false positives. However, for the apps with relatively unique aliases, our method achieves a high precision.

6.4 Evaluation on Opinion Summation

We then evaluate the accuracy for comparative opinion summation, first on how accurate can we find the correct topics in each comparative review. We run our topic analysis algorithm on all identified reviews to extract the

Table 6. Precision of comparative review identification. *Note that we select 150 identified reviews of the top 5 apps for each category, and the correct reviews are manually labeled.*

Category	#Examined	#Correct	Precision
Tools	150	132	88.0%
Photography	150	148	98.7%
Communication	150	148	98.7%
Media&Video	150	135	90.0%
Social	150	143	95.3%
Transportation	150	137	91.3%
Overall	900	843	93.7%

Table 7. Precision of opinion summation. *For each row,* #Total *is the number of reviews that are extracted which contains the corresponding topic;* #Examined *is the number of reviews we have manually checked; The* Correct-Topic *column is the number and percentage of the reviews whose topics are correctly extracted among all examined reviews; The* Correct-Pref *column is the number and percentage of the reviews among all correct-topic reviews in which the preferred apps are correctly identified.*

Topic	#Total / #Examined	Correct-Topic	Correct-Pref	Overall
Ad/Spam	317 / 21	13 / 61.9%	8 / 61.5%	38.1%
Battery	390 / 48	30 / 62.5%	25 / 83.3%	52.1%
Memory	394 / 26	19 / 73.1%	17 / 89.5%	65.4%
Performance	1023 / 84	65 / 77.4%	59 / 90.8%	70.2%
Price	187 / 25	22 / 88.0%	18 / 81.8%	72.0%
Security	75 / 4	4 / 100.0%	4 / 100.0%	100.0%
Stability	532 / 44	32 / 72.7%	24 / 75.0%	54.5%
UI	557 / 45	40 / 88.9%	34 / 85.0%	75.6%
Usability	585 / 68	66 / 97.1%	52 / 78.8%	76.5%
Overall	4060 / 365	291 / 79.7%	241 / 82.8%	66.0%

topics included in each review. The topics considered in this experiment are still the 9 common topics listed in Table 3.

Table 7 shows the results of topic identification on the 10,499 reviews we have identified from the last step. Here we listed the number of reviews containing each topic. In total, we have identified 3,130 reviews that having at least one of the 9 topics. The total number of topics identified are over 4,000 in Table 7. The number of topics (4,060) is greater than the number of identified reviews (3,130) because some reviews many contain more than one topic. For example, a user may compare both battery consumption and performance of two apps in one review.

We use the topics identified for the 900 reviews from Table 6 to evaluate the accuracy of topic identification. The result is shown in Table 7. Out of the 900 reviews, we found 290 of them with 365 different comparative opinions (an opinion is a topic with a preference). Similar to the evaluation on comparative review identification, we have recruited three volunteers to examine the correctness of topic and preference for all 365 opinions. We also take the majority opinion when they did not agree with each other.

Upon manual inspection, about 80% of the topics are correctly identified. Particularly, the precision for "Battery" and "Ad/Spam" is relatively low because some app names in the Tools category contain topic-related keywords such as "battery" (for example *"Battery Doctor"*) and "ad" (for example *"Ad Blocker"*). When these app names are

used in the reviews, we incorrectly identified them as battery-related or advertising topics. This is one of the limitations of our method, but the issue could be mitigated by ignoring certain app names during topic analysis.

We then conduct preference analysis on all the topics, each topic was attached with an opinion on whether it is preferred based on the specific topic. We evaluate the accuracy of preference analysis on the correct topics from Table 7 also with three volunteers (similar as above). The result shows that we can correctly identify the preferred app for 82.8% of the topics.

The overall precision, which is calculated as the product of topic analysis precision and preference analysis precision, is 66.0%. Although the precision can still be potentially improved, we think it provides an acceptable result for common usage scenarios. Many users would read the reviews by themselves when they compare apps, thus the high precision of comparative review identification and topic analysis is enough for them to locate the appropriate reviews. For example, when a user wants to find a taxi booking app cheaper than Uber, although we cannot give the precise answer directly, we can return many reviews comparing the price with Uber, which can help the user find the desired answer. Even if some users might not read the reviews by themselves, our method can still be used to identify which app is preferred by most reviewers with an acceptable precision.

7 DISCUSSIONS

7.1 Limitations

Indirect comparisons. We have considered only direct comparisons in the reviews in this paper. Thus some comparative relations might be missed by our approach. For example, if a user tried two similar apps and wrote reviews for both apps but did not mention each other in both reviews, that would be an indirect comparison. We are unable to find these kinds of indirect comparisons because we only collected a subset of reviews for each app, thus it is difficult to find the reviews written by the same user. We believe indirect comparison could be interesting for a market-scale analysis.

Accuracy of comparative sentence mining. Mining comparative sentences from reviews is an important research topic beyond mobile app user reviews. However, there are no mature approaches despite the community interests on the topic [10, 11, 14, 19]. We have improved the comparative sentence identification significantly by including the app name aliases in our method. However, more investigation is needed because identifying comparative sentences still faces many challenges in NLP.

Newly-released apps. Some newly-released apps could have competitive relations with existing apps, however, our method is unable to find the relations due to the low number of user reviews. These apps are not our focus because our method is based on the assumption that when a user wants to find a better app, there is someone with similar preferences who have already tried other apps and written reviews.

Apps with generic names. Our method is not accurate enough when identifying reviews referencing the apps with generic names (such as *task killer*, *wifi analyzer*, etc.), because we do not know whether the terms appeared in the reviews are referencing the apps or just their normal meanings. We can still extract comparative opinions for this kind of apps by analyzing the reviews that belong to these apps (where the apps are the *host* in comparisons). However, it would be interesting to determine whether a generic term is referencing a specific app using some context-aware NLP techniques. Similarly, when app names contain topic-related keywords, the opinion analysis accuracy is also severely impacted.

7.2 Future Work

Precision improvement. The precision of our method can be potentially improved by optimizing each step of the method. First, we can find more accurate aliases and summarize more accurate comparative patterns by manually checking a large amount of comparative reviews. Then we can improve the precision for both topic analysis and preference analysis by manually excluding general names that cause a lot of false positives. Finally,

as the amount of comparative reviews is small compared to the number of users, we can use crowd-sourcing techniques to identify incorrectly extracted comparative opinions. As this may involve heavy manual efforts or crowd-sourcing support, we will explore this direction in our future work.

Market-scale analysis. As we can see from the results, the number of comparative opinions identified tends to be small even for very popular apps. This is because the limitation of our dataset as we have crawled at most 4,000 reviews for each app. We expect that a market-scale analysis will generate much more comparative opinions and make the results more meaningful. The goal of this paper is to demonstrate the potential effectiveness of our proposed approach, while a complete market-scale evaluation is beyond our reach as we cannot expect to crawl all reviews for more than two millions apps.

Topic distribution. One interesting phenomenon we noticed in topic analysis is that the topics discussed in user reviews are not evenly distributed. For example, many users talk about UI and crashes while very few users talk about privacy and security. The reasons may include the unawareness and difficulty for users to discover privacy and security issues. We believe analyzing the topic distribution in a finer granularity would help us understand users' expectation, which might be helpful for app development and app recommendation.

Tool support and user studies. Our method enables some interesting applications based on app comparisons. For example, a comparative review search engine can recommend better apps based on an existing app and user preferences, and it can also provide fine-grained comparisons between two apps. We are working on designing and implementing such tools, and it is also worthwhile to evaluate and improve the tools based on user studies.

8 CONCLUDING REMARKS

This paper proposes a method to automatically summarize comparative opinions from large-scale user reviews, which can be used to provide more fine-grained app comparisons. The method includes a comparative review identification step based on alias matching and comparative pattern matching, and an opinion summation step based on topic analysis and preference analysis. According to a set of experiments on around 5 million reviews of more than 2,000 popular apps in 6 categories from Google Play, we are able to summarize comparative opinions from large-scale user reviews with an acceptable accuracy.

Along this direction, we believe a market-scale study that explores the comparative opinions in all user reviews in markets may extract more detailed comparisons between apps and reveal more interesting applications based on the comparisons.

ACKNOWLEDGMENTS

The authors would like to thank all developers contributed to the project and all volunteers involved in the experiments. The work is supported in part by the National Key Research and Development Program 2016YFB1000801 and the National Natural Science Foundation of China under Grant No. 61421091.

REFERENCES

[1] LLC Blanc Media. 2017. The Sweet Setup: We recommend the best apps for your iPhone, iPad, and Mac. (2017). Retrieved July 1, 2017 from http://thesweetsetup.com/

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

[3] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace. In *ICSE 2014*. 767–778. https://doi.org/10.1145/2568225.2568263

[4] CMU CHIMPS Lab. 2014. PrivacyGrade: Grading The Privacy Of Smartphone Apps. (2014). Retrieved May 1, 2017 from http://privacygrade.org/

[5] Murthy Ganapathibhotla and Bing Liu. 2008. Mining Opinions in Comparative Sentences. In *COLING '08*. 241–248. http://dl.acm.org/citation.cfm?id=1599081.1599112

[6] Emitza Guzman and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*. 153–162.
[7] Thomas Hofmann. 1999. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*. Morgan Kaufmann Publishers Inc., 289–296. http://dl.acm.org/citation.cfm?id=2073796.2073829
[8] Claudia Iacob, Varsha Veerappa, and Rachel Harrison. 2013. What Are You Complaining About?: A Study of Online Reviews of Mobile Applications. In *Proceedings of the 27th International BCS Human Computer Interaction Conference (BCS-HCI '13)*. Article 29, 6 pages. http://dl.acm.org/citation.cfm?id=2578048.2578086
[9] Reyhaneh Jabbarvand, Alireza Sadeghi, Joshua Garcia, Sam Malek, and Paul Ammann. 2015. EcoDroid: An Approach for Energy-based Ranking of Android Apps. In *Proceedings of the Fourth International Workshop on Green and Sustainable Software (GREENS '15)*. 8–14. http://dl.acm.org/citation.cfm?id=2820158.2820161
[10] Haojian Jin, Tetsuya Sakai, and Koji Yatani. 2014. ReviewCollage: A Mobile Interface for Direct Comparison Using Online Reviews. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI '14)*. 349–358. https://doi.org/10.1145/2628363.2628373
[11] Nitin Jindal and Bing Liu. 2006. Identifying Comparative Sentences in Text Documents. In *SIGIR '06*. 244–251. https://doi.org/10.1145/1148170.1148215
[12] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E Hassan. 2015. What do mobile app users complain about? *IEEE Software* 32, 3 (2015), 70–77.
[13] Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25, 2-3 (1998), 259–284.
[14] Shasha Li, Chin-Yew Lin, Young-In Song, and Zhoujun Li. 2013. Comparable entity mining from comparative questions. *IEEE Transactions On Knowledge And Data Engineering* 25, 7 (2013), 1498–1509.
[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[16] Dennis Pagano and Walid Maalej. 2013. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE 2013)*. 125–134.
[17] Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Found. Trends Inf. Retr.* 2, 1-2 (Jan. 2008), 1–135. https://doi.org/10.1561/1500000011
[18] Dae Hoon Park and Catherine Blake. 2012. Identifying Comparative Claim Sentences in Full-text Scientific Articles. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse (ACL '12)*. 1–9. http://dl.acm.org/citation.cfm?id=2391171.2391173
[19] Dae Hoon Park, Mengwen Liu, ChengXiang Zhai, and Haohong Wang. 2015. Leveraging User Reviews to Improve Accuracy for Mobile App Retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. 533–542. https://doi.org/10.1145/2766462.2767759
[20] Yonggang Qiu and Hans-Peter Frei. 1993. Concept Based Query Expansion. In *SIGIR '93*. 160–169. https://doi.org/10.1145/160688.160713
[21] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, Vol. 1631. 1642.
[22] Rajesh Vasa, Leonard Hoon, Kon Mouzakis, and Akihiro Noguchi. 2012. A Preliminary Analysis of Mobile App User Reviews. In *Proceedings of the 24th Australian Computer-Human Interaction Conference (OzCHI '12)*. 241–244. https://doi.org/10.1145/2414536.2414577
[23] Phong Minh Vu, Tam The Nguyen, Hung Viet Pham, and Tung Thanh Nguyen. 2015. Mining User Opinions in Mobile App Reviews: A Keyword-Based Approach (T). In *ASE 2015*. 749–759.
[24] Wikipedia. 2017. App Store. (2017). Retrieved July 1, 2017 from https://en.wikipedia.org/wiki/App_Store_(iOS)
[25] Wikipedia. 2017. Google Play. (2017). Retrieved July 1, 2017 from https://en.wikipedia.org/wiki/Google_Play
[26] Jinxi Xu and W. Bruce Croft. 1996. Query Expansion Using Local and Global Document Analysis. In *SIGIR '96*. 4–11. https://doi.org/10.1145/243199.243202
[27] Kaiquan Xu, Stephen Shaoyi Liao, Jiexun Li, and Yuxia Song. 2011. Mining comparative opinions from customer reviews for Competitive Intelligence. *Decision support systems* 50, 4 (2011), 743–754.
[28] LLC. PCMag Digital Group Ziff Davis. 2017. PCMAG: Reviews of mobile apps. (2017). Retrieved July 1, 2017 from http://www.pcmag.com/reviews/mobile-apps

Received February 2017; revised May 2017; accepted July 2017