Deriving User Preferences of Mobile Apps from Their Management Activities

XUANZHE LIU, Peking University
WEI AI, University of Michigan
HUORAN LI, Peking University
JIAN TANG, University of Michigan
GANG HUANG, Peking University
FENG FENG, Wandoujia Lab
QIAOZHU MEI, University of Michigan

App marketplaces host millions of mobile apps that are downloaded billions of times. Investigating how people manage mobile apps in their everyday lives creates a unique opportunity to understand the behavior and preferences of mobile device users, infer the quality of apps, and improve user experience. Existing literature provides very limited knowledge about app management activities, due to the lack of app usage data at scale. This article takes the initiative to analyze a very large app management log collected through a leading Android app marketplace. The dataset covers 5 months of detailed downloading, updating, and uninstallation activities, which involve 17 million anonymized users and 1 million apps. We present a surprising finding that the metrics commonly used to rank apps in app stores do not truly reflect the users' real attitudes. We then identify behavioral patterns from the app management activities that more accurately indicate user preferences of an app even when no explicit rating is available. A systematic statistical analysis is designed to evaluate machine learning models that are trained to predict user preferences using these behavioral patterns, which features an inverse probability weighting method to correct the selection biases in the training process.

CCS Concepts: • **Information systems** → *Mobile information processing systems*; • **Human-centered computing** → Empirical studies in ubiquitous and mobile computing;

Additional Key Words and Phrases: Mobile apps, app management activities, behavior analysis

ACM Reference Format:
Xuanzhe Liu, Wei Ai, Huoran Li, Jian Tang, Gang Huang, Feng Feng, and Qiaozhu Mei. 2017. Deriving user preferences of mobile apps from their management activities. ACM Trans. Inf. Syst. 35, 4, Article 39 (July 2017), 32 pages.
DOI: http://dx.doi.org/10.1145/3015462

X. Liu and W. Ai made equal contribution to this work.
This work was supported in part by the High-Tech Research and Development Program of China under grant 2015AA01A203 and the Natural Science Foundation of China under grants 61370020, 61421091, and 61528201. The work of Q. Mei and W. Ai was supported in part by the National Science Foundation under grant IIS-1054199 and an MCubed grant from the University of Michigan.
Authors' addresses: X. Liu, H. Li, and G. Huang, Key Lab of High-Confidence Software Technology, Peking University, No. 5 Yiheyuan Rd., Haidian District, Beijing, P. R. China, 100871; emails: {liuxuanzhe, lihuoran, hg}@pku.edu.cn; W. Ai, J. Tang, and Q. Mei, School of Information, University of Michigan, 105 S. State St., Ann Arbor, MI 48109; emails: aiwei@umich.edu, tangjianpku@gmail.com, qmei@umich.edu; F. Feng, Wandoujia Lab, B2, Dongsheng Science & Technology Park, Haidian District, Beijing, P. R. China, 100192; email: jackfeng@wandoujia.com.

1. INTRODUCTION

The prevalence of smartphone applications (also known as apps) has introduced an evolutionary change to people's everyday lives. By 2014, Apple App Store and Google Play each hosted more than 2 million apps that had been downloaded more than 100 billion times.[1–3] Many apps are also hosted through other marketplaces, such as Amazon App Store, Samsung App Store, F-Droid,[4] and Fetch.[5] In regions where the native app marketplaces are inaccessible, these third-party marketplaces play a major role in facilitating users to find, download, and manage mobile apps.

To help users find and explore high-quality apps, most app marketplaces gather user-input ratings of apps in the format of like/dislike votes, numerical ratings, or free-text comments. Much work has been done to analyze these ratings [Liu 2012]. However, among the hundreds of millions of users who downloaded and used these apps, only a tiny proportion have left reviews [Lim et al. 2015]. Although popular apps are rated heavily, the majority of apps, especially new apps, receive very few or even no ratings, making potentially high-quality apps invisible from the users. Biased, fake, paid-off, and malicious reviews also commonly exist, which compromises the credibility of online ratings as the indicator of users' true preferences of apps [Ott et al. 2012].

We believe a more credible and more robust indicator of user preferences can be derived from a different signal, the signal of users voting an app using their feet. We hypothesize that users who like an app manage it differently from those who do not. Users' preference toward an app can be revealed by the way they manage the app, and these actions are just like the votes made by the users. Indeed, it is reported that 80% to 90% of mobile apps were downloaded, used only once, and then deleted by a user.[6] These users, although few of them bother to leave a review, are clearly not in favor of the app. However, other users are much more engaged, who download an app, keep it up to date with the updates, and immediately install it back after they get a new device or update the operating system.

More generally, *app management activities* can include searching, downloading, installing, updating, and uninstalling an app. They can provide much richer and less biased indicators of whether the user actually prefers the app or not. In other words, if one can infer user preferences from the patterns among these app management activities, one could provide an accurate and unbiased indicator of the quality of apps even if they are not rated by any user. Many marketplaces count how many times an app has been downloaded (or how many users have downloaded the app), hoping to provide evidence of the quality of the app when the ratings are rare. These simple counts, however, may not really reflect true user preferences when they are interpreted out of context—the actual sequences of user activities (e.g., a user may download an app but discard it right away). Such a metric can also be easily manipulated, for example, through fake users or repeated downloads.

A deeper understanding of how user preference and app quality can be inferred from user activities is critical. Apparently, this requires analyzing the activity sequences of a large number of real users managing a large number of apps, a dataset that does not exist in the literature. Indeed, previous studies rely on datasets of user activities collected from small groups of volunteers, usually limited to hundreds of users and

[1]http://ipod.about.com/od/iphonesoftwareterms/qt/apps-in-app-store.htm.
[2]http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.
[3]http://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store/.
[4]http://f-droid.org/.
[5]Fetch has been part of BlueStack: http://www.bluestacks.com.
[6]http://www.digitaltrends.com/mobile/16-percent-of-mobile-userstry-out-a-buggy-app-more-than-twice/.

devices [Rahmati and Zhong 2013; Rahmati et al. 2012] due to the inability to collect such large-scale behavioral data.

Recent developments of app marketplaces have opened the channel to collect user activities on a much larger scale. Many marketplaces provide their own native smartphone apps through which their users can manage apps installed on their devices. Through such a tool, sequences of app management activities are collected from millions of users. Even though only particular types of user activities are recorded (e.g., downloading, updating, and uninstalling apps), such large-scale collections of behavioral data already present brand-new opportunities to analysts. Knowledge discovered from these datasets provides not only a better understanding of the behavior of mobile users but also insights for the app marketplaces to assess the quality of apps and to provide effective recommendations to their users.

We present the first analysis of app management activities at the scale of millions of Android users.[7] The behavioral data are collected through a leading app marketplace in China called *Wandoujia*.[8] Wandoujia serves near half a billion registered users, 50 million of which are active on a daily basis. Developers can upload and publish their apps through Wandoujia, and consumers can search, download, update, install, uninstall, and rate apps through Wandoujia's native app. The behavioral data studied in this article cover timestamped activities of downloading, updating, and uninstalling more than 1 million apps by more than 17 million anonymized users, spanning 5 months (May 1, 2014 to September 30, 2014). Based on this largest dataset to date, we are able to conduct a comprehensive study of how users manage mobile apps and how their preferences can be inferred from these activities.

In summary, we present the following contributions:

—The first empirical analysis of app management activities collected from millions of users. We characterize how 17 million Android users manage their mobile apps.
—A descriptive analysis of how app management activities correlate with user-input ratings of these apps. Surprisingly, we find that more downloads do not indicate higher ratings, especially when explicit ratings are rare.
—We identify sequential patterns of user management activities that are actually correlated with online ratings of apps, which can be used as better indicators of user preference and app quality.
—We present a systematic evaluation of how well the behavioral indicators from app management activities can be combined with machine learning algorithms to predict the quality of apps.
—We introduce a principled statistical method—inverse probability weighting (IPW)—to correct selection biases in the preceding analysis. The new method, borrowed from economics and statistics, not only improves the prediction performance but also provides robust interpretation of the model. The method can be widely applied to data analysis tasks where selection biases exist.

Part of the results were reported in our recent work at WWW 2016 [Li et al. 2016]. For those who have read the conference version, this extension includes a comprehensive literature review and many more details and discussions about data collection and data analyses, as well as empirical findings. In particular, we describe how the data

[7]Our study has been approved by the research ethics board of the Institute of Software, Peking University. The data are legally used without leaking any sensitive information. A sample of the dataset has been released, along with our previous work [Li et al. 2015b], which can be accessed at http://www.sei.pku.edu.cn/∼liuxzh/appdata. For more information, please contact liuxuanzhe@pku.edu.cn.
[8]http://www.wandoujia.com/.

collection is carefully done to preserve privacy and research ethics. The introduction and application of IPW (Section 6) are completely new.

The rest of the article is organized as follows. We first relate our study to existing literature in Section 2. Section 3 describes the dataset and presents a descriptive analysis of app management activities. Section 4 identifies patterns of app management activities that are correlated with the user ratings of apps. Section 5 presents a prediction analysis to measure how much these activity patterns improve the ranking of apps. We then introduce how to use IPW to correct selection biases in Section 6. We discuss implications and potential limitations of the work in Section 7 and conclude in Section 8.

2. RELATED WORK

To the best of our knowledge, this is the first empirical study on app management activities at the scale of millions of users and millions of apps. Our work is generally related to the literature of analyzing user preferences and assessing app quality. Recently, analyzing behavioral data collected from smartphones or app stores has attracted much attention. According to the type of data being analyzed, we can categorize existing literature into four aspects.

2.1. User Reviews in App Stores

Like other online reputation systems, app stores such as Apple App Store and Google Play have accumulated a large volume of public user reviews. Numerical ratings, either binary or in multiple scales, directly measure the popularity and quality of an app. Textual reviews usually reflect the sentiment of app users and their preferences of apps. Goul et al. [2012] analyzed sentiment in 5,000 reviews to facilitate requirements engineering. In a larger study, Chandy and Gu [2012] analyzed 6,319,661 reviews from Apple App Store for spam classification. Harman et al. [2012] conducted data analysis on technical, nontechnical, and business attributes extracted from user reviews on the Blackberry store.

A large body of work appears more recently. Iacob and Harrison [2013] produced a system that automatically extracts feature requests and bug reports from app reviews. Fu et al. [2013] presented WisCom, a system that analyzes millions of user ratings and narrative comments from app stores. Chen et al. [2014] presented AR-Miner, which collects narrative user reviews and groups them through topic modeling. AR-Miner prioritizes the narrative reviews using a ranking scheme. Park et al. [2015] utilized user reviews to improve mobile app retrieval by designing a topic model, AppLDA, which discards review-only topics. This enables developers to inspect the reviews that discuss features present in the app descriptions. The authors tested the topic model on 1,385,607 reviews mined from 43,041 apps. Khalid et al. [2015] manually categorized 6,390 negative reviews from a sample of 20 free iOS apps and reported the most frequent causes of complaints. Overall, the apps combined more than 250,000 reviews, and so 6,390 reviews is a statistically representative sample at the 95% confidence level. Maalej and Nabil [2015] proposed a classification approach to identify bug reports and feature requests from user reviews. They found that multiple binary classifiers outperform a single multiclass classifier. In particular, it is surprising that some commonly used NLP techniques, such as stopword removal and lemmatization, can negatively affect the performance of classification. Panichella et al. [2015] presented a system that automatically classifies user reviews based on a predetermined taxonomy to support software maintenance and requirements evolution. The authors manually labeled 1,421 sentences extracted from reviews and reported 0.85 precision and 0.85 recall when the system is trained using language structure, content, and sentiment features. Palomba et al. [2015] devised an approach named *CRISTAL* to guide developers to improve their apps based on informative crowd reviews. Tian et al. [2015] investigated how high-rated apps are different from low-rated apps and summarized 28 factors that can possibly increase an app's ratings. McIlroy et al. [2016] presented a study on app updates in Google Play and proposed to label the types of user issues raised in user reviews. Villarroel et al. [2016] implemented a tool called *CLAP*, which automatically clusters user reviews and recommends the cluster of reviews that should be prioritized.

Although user reviews have been widely adopted, assessing the quality of apps and learning a user's preference based on only user reviews suffer from data sparseness, ambiguity, and validity—for example, some apps gain very few reviews, and some reviews convey unclear opinions [Lim et al. 2015].

2.2. App Usage Logs

A less biased way to evaluate an app is through observing how the app is actually used by real users. Researchers build auxiliary apps to monitor the activities and performance of other apps installed on the same device. Ravindranath et al. [2012] developed AppInsight, a system that instruments mobile app binaries to automatically identify and characterize the critical paths in user transactions across asynchronous-call boundaries. It logs app usage data such as launching frequency, traffic volume, and access time. The data can provide sufficient information to infer user preferences and interests. However, such a "monitor" app is usually deployed as a system-level service [Agarwal et al. 2010]. Not many users are willing to voluntarily install them on their devices, which creates a major obstacle to conducting this type of research at scale. Although some widely deployed commercial apps such as Flurry[9] and PreEmptive[10] can also be used to monitor app usage, no evidence shows that external researchers can access such data and conduct scientific research. At a small scale, Shi and Ali [2012] analyzed app usage of 100,000 users collected by the GetJar system to provide personalized recommendations.

Essentially, the data used in our study are also collected through the log of Wandoujia's management app. The app usage log collected through Wandoujia is by far the largest in the literature, covering millions of users and millions of apps (a sample of the data has been released for public research [Li et al. 2015b]). Many interesting research problems can be pursued based on such a dataset. For example, Lu et al. [2016] proposed the PRADA method to predict the most popular device models for a given app. The study presented in this article mainly uses the app management activities in the Wandoujia dataset.

2.3. Observational Data in Field Studies

Another commonly used approach to observing and understanding user behaviors is field study. A few field studies were conducted based on the LiveLab datasets [Tossell et al. 2012; Rahmati and Zhong 2013; Rahmati et al. 2012]. Rahmati and Zhong [2013] conducted a longitudinal study to collect the network usage data of 24 users, which were used to analyze how users use the network on their smartphones. They also collected actual usage data from 14 teenage smartphone users in a 4-month field study. The same group of authors presented another study that involved 34 iPhone 3GS users and reported how users with different economic backgrounds use smartphones differently [Rahmati et al. 2012]. Many other analyses have been done based on similar fields studies, which reported the diversity of cellular usage from different user groups, different app categories, and different OS platforms [Jung et al. 2012; Chittaranjan et al. 2013; Böhmer et al. 2011; Böhmer and Krüger 2013; Falaki et al.

[9]http://www.flurry.com/.
[10]http://www.preemptive.com/.

2010]. For example, similar apps (e.g., weather apps) may have a significant difference in generating network traffic [Sani et al. 2013]. In most of these field studies, devices have to be provided to human subjects. Similar to collecting usage data from instrumented devices, these field studies normally only involve a small group of subjects and are hard to scale.

Our work is distinguished from existing work in two aspects. On one hand, we investigate a particular signal that reflects the users' attitudes toward an app, such as activities of downloading, updating, and uninstalling apps. Such activities are more objective indicators compared to user-input reviews. On the other hand, we analyze collective and longitudinal activities from millions of users spanning 5 months. The data we study have a much larger scale than other app usage data sets in the literature, have much less bias than user reviews, and involve a significantly larger user population than the field studies. The availability of such a dataset provides a unique perspective of understanding user preferences of smartphone apps.

2.4. Behavior Sequences and Log Analysis

Our work is also related to the general literature of analyzing user behavior logs, such as search and browsing logs [Jansen 2008; Jiang et al. 2013], trajectory logs [Zheng and Zhou 2011], and other types of activity logs [Hu et al. 2008]. Like our work, many explorations on these data also involve sequential data analysis and prediction models. We present the first large-scale analysis of app management logs. Compared to search engine logs, user preferences in app management activities are implicit and have to be inferred carefully. However, users' app management activities are less sensitive to environment-imposed biases, such as the position biases of search results and the location biases of user trajectories. The method that we introduce for correcting selection biases can also be applied to these domains.

3. A DESCRIPTIVE ANALYSIS

We start by describing the data used in this study. For the completeness of presentation, we include some results that have already been reported in our recent poster paper [Li et al. 2015a].

3.1. The Wandoujia Dataset

Founded in 2009, Wandoujia has grown to be a leading Android app marketplace in China. Up to 2014, Wandoujia has hosted more than 1.5 million apps and has been accessed by more than 200 million Android devices. Users can access Wandoujia via its Web site, its Windows client, and its native management app. The Wandoujia management app works as a system service and functions just like Apple App Store and Google Play. Through the app, a user can conduct various activities to manage the apps on their devices, such as browsing and searching apps, installing and uninstalling apps, and checking and installing updates of apps. Upon the permission of users, the Wandoujia management app logs these management activities and uploads the logs on its own private and secure server.

In our study, we collected app management activities of Wandoujia users spanning 5 months from May to September 2014, covering more than 17 million users (actually devices) and 1 million apps. A timestamped record is logged whenever a user downloads, updates, or uninstalls an app via the Wandoujia management app. The total numbers of users, apps, and activity log entries involved in our study are reported in Table I. There is an average of 227.6 management activities collected per app, which is 13.9 per device. In addition to app management activities, we also collected the complete set of online ratings posted on Wandoujia, with an average of 4.01 per app.

Table I. Summary Statistics of the Data

Users (Devices) (#)	17,303,122
Apps (#)	1,054,969
Activities (#)	240,108,930
Online Ratings (#)	4,225,153

Note that similar to Apple App Store and Google Play, the Wandoujia management app records the activities of only the apps that are actually installed, and only the activities that are conducted through the Wandoujia app. Therefore, we do not distinguish between the activities of "downloads" and "installations." When an app is directly downloaded from the Web and installed (i.e., not through the Wandoujia app), this downloading activity is not logged. Similarly, only updating and uninstallation actions conducted through the Wandoujia app are logged. Therefore, our dataset may miss some apps or management activities of certain apps if they are not conducted through the Wandoujia management app.

It is also possible that a user's activity sequence spans longer than 5 months. There may be incomplete activity sequences of certain users and certain apps in our dataset. Nevertheless, given the longitudinal collection and the very large volume of activities, we anticipate that these possible limitations would not have a significant effect on our analyses. In the following, we present some characteristics of app management activities in this dataset.

Indeed, the usage information of the apps can also be very important, such as how often is the app used, as it provides insight into an app's quality. However, such information is not captured by the Wandoujia management app. In addition, the goal of this work is to explore the correlation between management activities and app qualities, so the usage information is not included in our work.

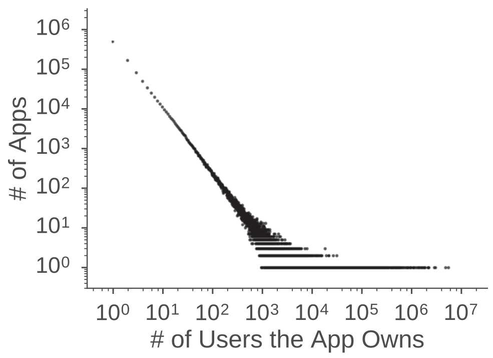
3.2. Ethical Consideration

Undoubtedly, collecting and analyzing user behavioral data require careful consideration of research ethics. We took a series of steps to preserve the privacy of involved users in our dataset. First, all raw data collected for this study are kept within Wandoujia's data warehouse servers, which are placed behind the company's firewall. Second, our data collection logic and analysis pipelines were completely governed by three Wandoujia employees to ensure compliance with the commitments of Wandoujia privacy policy in the Terms-of-Use statements. Finally, Wandoujia employees[11] took charge of anonymizing the user identifiers. The dataset includes only timestamped activities and no personal information for the users covered by our study period. No individual user can be identified.

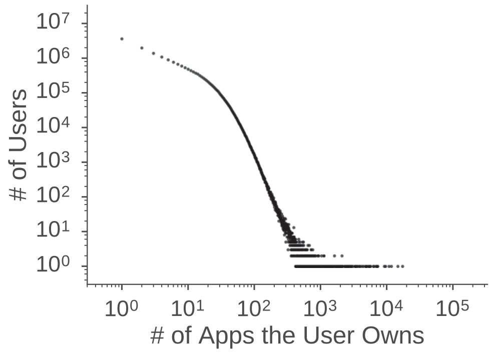
3.3. Distributions of Management Activities

We first report a series of distributions that may help validate the representativeness of the data. We start with the distributions of the popularity of apps. The popularity of an app can be measured using either the total number of downloads, the most recent downloads (e.g., in the past week or month), or the total number of users. When an app is rarely rated, a marketplace usually provides such a popularity measure as an indicator of the quality of the app. Figure 1(a) plots the distribution of the number of users per app (i.e., the number of devices on which at least one activity of that app has been logged). The popularity of apps follows a typical power law distribution on this log-log plot. More than 95% apps are downloaded by fewer than 1,000 devices. A few apps are downloaded by millions of devices. This distribution is consistent with

[11]One co-author of this paper, Feng Feng, is the co-founder and current CTO of Wandoujia.



(a) Distribution of users per app.



(b) Distribution of apps per user.

Fig. 1. Distribution of app popularity. (a) The x-axis indicates the number of users of an app, and the y-axis indicates the number of apps that are installed by a certain number of users. App popularity follows a typical power law distribution. (b) The x-axis indicates the number of apps installed by a user, and the y-axis indicates the number of users who installed a certain number of apps. The distribution follows power law in the tail.

observations in many other behavior data at the population level [Barabási and Albert 1999], which usually indicate an effect of "rich get richer." Indeed, the highly ranked apps may attract even more users, whereas barely rated apps are buried in the long tail. Similarly, the number of ratings an app receives also presents a typical power law distribution, which we omit here for brevity.

Figure 1(b) plots the distribution of the number of apps that a user installs on her device. Intuitively, many users use only a few apps, whereas a few others try out a large number of apps. The distribution obeys a power law in its tail distribution, which is also a typical phenomenon of user behaviors [Newman 2005]. We would like to point out that the number of apps installed on a device is likely to be underestimated, as lots of devices have preloaded apps and users might install apps through channels other than Wandoujia. This might have made the distribution not follow one power law distribution over the entire range.

We then look at how the users manage their apps throughout a day. Figure 2 plots the percentages of user activities over 24 hours of a day. Common sense assumes that a download and an updating activity reflect a user's preference toward an app, and an uninstallation indicates that the user is not in favor of the app. We thus aggregate downloading and updating activities (solid blue line) in this plot and compare them with uninstallations (dashed green line). All timestamps are converted to Beijing time (UTC+8), as most Wandoujia users are in China.

As one may anticipate, app management activities start to increase sharply from 7 a.m. and stay high throughout the day. Both downloading/updating and uninstallation activities fall around lunch and dinner. It is interesting to observe clear peaks of both types of activities, indicating that most users may have a routine schedule for managing their apps.

To further investigate whether the users do have a routine schedule of app management, we plot the distribution of the time intervals between any two consecutive activities of the same user (Figure 3). Most consecutive activities are conducted within less than an hour (the leftmost data point), which is not surprising. These are likely to be activities conducted in the same session (e.g., updating a batch of apps). However, when the intervals are larger, we observe rather surprising patterns. There is a peak at every 24 hours, and between two peaks the time interval distributes as a reversed bell-shaped (either normal or Poisson) distribution. This suggests that a user does have

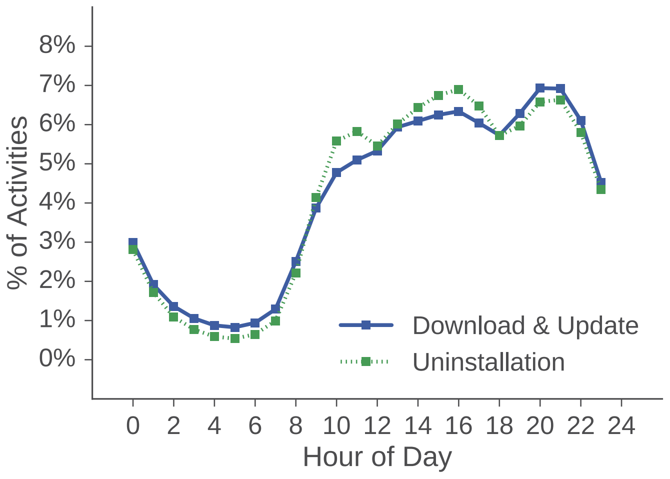


Fig. 2. Diurnal distribution of app management activities. Curves show the percentage of certain types of activities. Installations peak at prime television time; uninstallations present three peaks during the day.

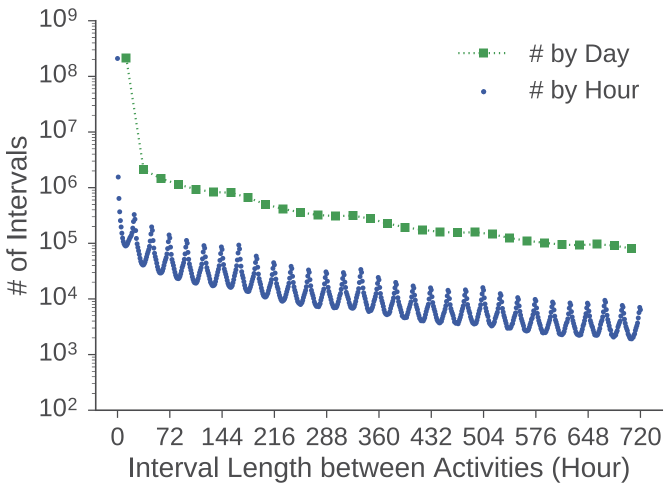


Fig. 3. Distribution of the length of time intervals between consecutive management activities. The blue curve presents the distribution by hour, and the green curve presents the distribution by day.

a routine time of day for housekeeping—she may not do it every day, but whenever she does it is likely to be around the same time of the day. If we plot the intervals by day instead of by hour, it follows an exponential distribution (the green line).

Many of the preceding results are consistent with common sense, indicating the validity and representativeness of our dataset. Some of them are a bit interesting. For example, the diurnal management activities are performed periodically, which indicates when users access the app marketplaces and the user requests distributions. Such a finding not only makes app marketplace operators optimize their bandwidth and server-side resources for fast content delivery but also provides insights on our following analysis of activity indicators (Section 4.2).

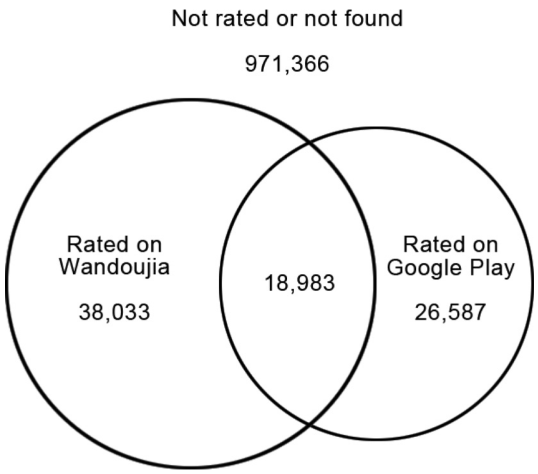


Fig. 4. Venn diagram of user-rated apps on Wandoujia and Google Play.

4. INFERRING USER PREFERENCES

We learned from Table I that the average number of management activities per app in 5 months is 50 times larger than the average number of ratings in 5 years. This motivates us to explore whether the abundant user activities could be a signal to infer the preferences of users and the quality of apps, either as a complement or a surrogate of the scarce user ratings. We start by demonstrating the limitations of user ratings.

4.1. Limitations of User Ratings

Intuitively, an app is considered to be of high quality if it is ranked highly by its users. In practice, however, there are quite a few limitations of directly using such a straightforward metric to assess app quality, as it can suffer from data sparseness and biases.

Most app marketplaces allow users to explicitly rate the apps they installed. For example, Google Play allows users to rate apps on a 1 to 5 scale where one star refers to the lowest quality and five stars refers to the highest. Wandoujia allows users to simply tag an app with “like” or “dislike.”

For comparison, we crawled user ratings of all apps from both Wandoujia and Google Play. In this way, given a specific app, we can synthesize its average ratings on both marketplaces. Figure 4 presents a Venn diagram of the apps that have at least one nontrivial management activity sequence (sequence with more than one activity) in our dataset, rated at the two marketplaces. Surprisingly, among the 1 million apps, only a small portion (less than 2%) is rated at both marketplaces. Others do not receive any rating on either one or two of the marketplaces.

For the 18,983 apps that are rated by the users at both marketplaces, we are able to compare their ratings. We explore the correlation of the numbers of ratings an app receives on the two marketplaces, which is plotted in Figure 5. Generally, the numbers of ratings per app at the two marketplaces are positively correlated. However, there are considerable biases, appearing as the noticeable vertical lines on the left and horizontal lines at the bottom of the plot. These data points refer to the apps that have many ratings on Google Play but very few ratings on Wandoujia, or the opposite. Some of these biases are likely caused by language or culture differences [Lim et al. 2015], whereas others can be introduced by regional barriers. It was reported that a local app can be very popular in specific areas [Xu et al. 2011] but not so popular in others. For

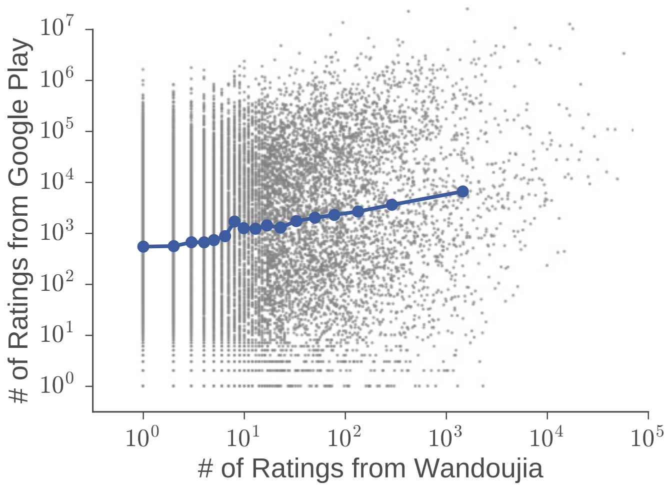


Fig. 5. Number of ratings that the apps received from both marketplaces. Each app is represented as a grey point in the scatter plot, and its *x/y* coordinates (in log scale) indicate the number of ratings from Wandoujia and Google Play, respectively. A binned curve is overlaid to show the correlation.

example, the Facebook app has 25,169,686 ratings on Google Play but only 1,644 on Wandoujia.

We then investigate whether the same app receives consistent ratings at the two marketplaces. We compute the average ratings of apps in both marketplaces and correlate them in a scatter plot. The average rating on Google Play is denoted as the *score*, and the average rating on Wandoujia is denoted as the *likerate*, computed as follows. These two notations will be used throughout the rest of the article.

$$\text{Score} = \frac{\sum \textit{stars}}{\text{number of ratings}} \tag{1}$$

$$\text{Likerate} = \frac{\text{number of } \textit{likes}}{\text{number of } \textit{likes} + \text{number of } \textit{dislikes}} \tag{2}$$

Figure 6 presents a positive correlation between the *scores* on Google Play and the *likerates* on Wandoujia for apps with at least five ratings on Wandoujia (7,513 in total). This indicates that user ratings are consistent overall at the two marketplaces. However, one can also easily identify many different or even contradictory ratings. The Pearson coefficient *r* of this correlation is 0.35, which becomes much smaller if we include apps with fewer ratings (0.30 for apps with at least three ratings, and 0.19 for all rated apps).

The comparison of user ratings at two marketplaces indicates that user ratings may be a trustful measurement of the quality of apps that have received many ratings, but it suffers from severe problems of sparseness and biases for apps that are rarely rated. Other types of review biases have been discussed in the literature. For example, it is reported that Asian users are less likely to commit their ratings [Lim et al. 2015]. There has been debate whether users who like or dislike an app are more likely to leave a review. In addition, there is also a prevalence of fake reviews [Ott et al. 2011] or review spams [Jindal and Liu 2008], as developers can manipulate the ratings by hiring the crowd to rate their apps (known as a water army in China). These problems have limited the effectiveness of users ratings as the sole indicators of app quality,

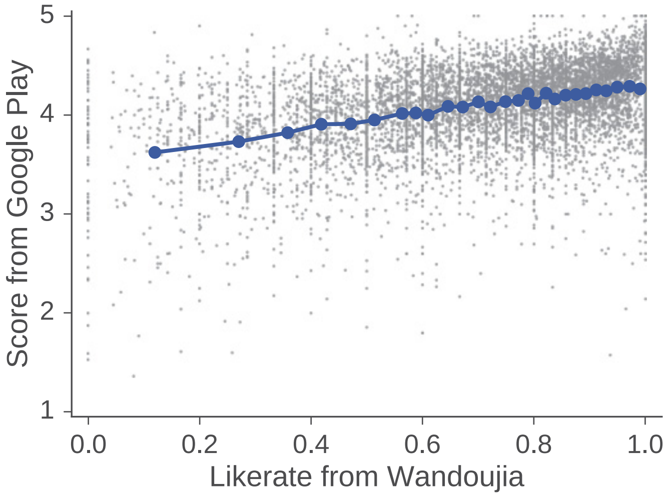


Fig. 6. Average ratings that the apps received from both marketplaces. Each app is represented as a grey point in the scatter plot, and its *x*/*y* coordinates (in log scale) indicate the likerates from Wandoujia and scores from Google Play, respectively. A binned curve is overlaid to show the correlation.

especially for those with fewer ratings. For these apps, it is critical to find alternative signals that better indicate their quality.

4.2. Activity Indicators

Most app marketplaces have recognized the limitation of user ratings. When the number of ratings is small, many marketplaces (e.g., Google Play) also report the total number of times an app is downloaded/installed, with the assumption that more downloads indicate better quality of the app. Some variants are also adopted, such as the number of users/devices that have downloaded the app (e.g., Wandoujia) or the number of most recent activities (e.g., Apple App Store). These metrics are generally extracted from the actual user activities instead of online ratings.

4.2.1. Popularity Indicators. Although how the users use an app intuitively implies how they like the app, can simple metrics such as the number of downloads reflect this preference? To verify this, we correlate the number of downloading activities to the likerate of an app, assuming that the likerate is a more objective measure of app quality when the ratings are abundant.

Motivated by the findings in Figure 6, we rank all apps that have received at least five ratings based on how many times they are downloaded in the 5 months, split them into equal-size bins, and plot the means of each bin in Figure 7. As the number of downloads follows a power law distribution, we plot the *x*-axis in log scale. A few observations can be made.

When the number of downloads is small (e.g., less than 1,000), it is negatively correlated with the average likerate. In other words, the more times an app is downloaded, the more likely it is *disliked* by users. This is rather counterintuitive and may occur for several reasons. Apps that are not frequently downloaded may be sensitive to fake "like" ratings, either artificially boosted by paid-off reviewers or maliciously rated down by their competitors. When the number of downloads exceeds 10,000, the correlation becomes positive but is still quite weak. In either case, this indicates that the number of downloads alone is at best a weak indicator of app quality, which may even be invalid for unpopular and new apps.

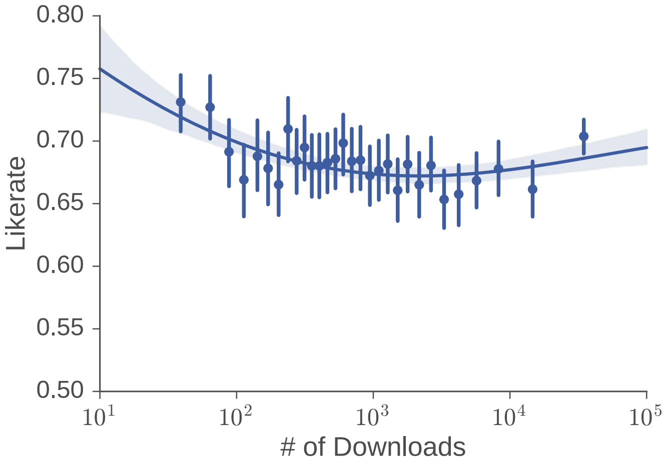


Fig. 7. Binned plot showing the correlation between the number of downloads and the likerate. All apps receiving five or more ratings are ranked with increasing numbers of downloads and split into equal-size bins. The mean likerate of each bin is plotted with a 95% confidence interval. A quartic regression model is also estimated and after transforming the number of downloads to log scale. Figures 8, 9, and 10 are plotted similarly.

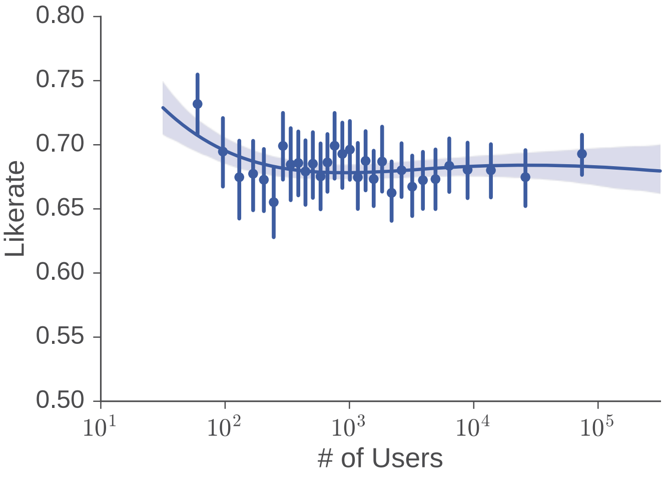


Fig. 8. Binned plot of the correlation between the number of users and the likerate.

We then explore whether the number of users/devices is more reliable. Unfortunately, it seems that such a metric is not a promising signal. From Figure 8, we see that the number of users (devices) installing an app is even negatively correlated with its likerate, although the correlation is quite weak.

Another intuitive assumption is that an uninstallation may indicate that the user disfavors an app. Although app marketplaces normally do not report this statistic, we can compare the number of uninstallations of an app to its user ratings. Instead of using the raw number of uninstallations, we introduce a metric called the D/U *ratio*, which is computed as the total number of downloads divided by the total number of uninstallations. Intuitively, the lower the D/U ratio, the more likely the app is disfavored. To validate this hypothesis, we plot the D/U ratio and the corresponding

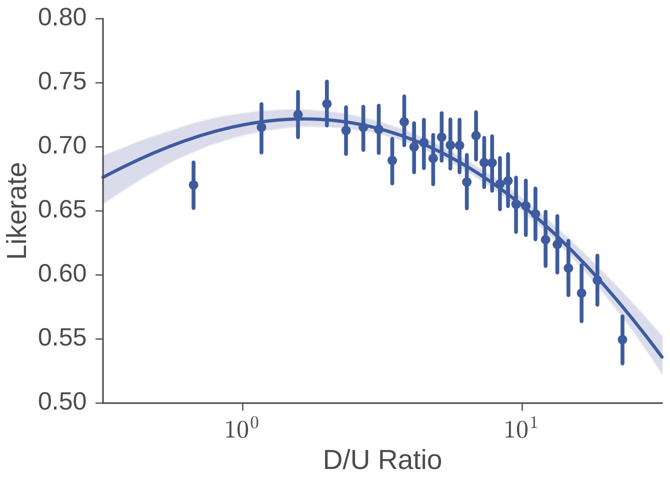


Fig. 9. Binned plot of the correlation between the likerate and the D/U ratio.

likerate in Figure 9. Again, we rank all apps with at least five ratings by D/U ratio and split them into equal-size bins.

Interestingly, when the D/U ratio is below 1, a positive correlation is observed between the D/U ratio and the likerate. When the D/U ratio is over 1, however, the correlation becomes negative. In either case, the correlation is rather weak and presents a long error bar. Apparently, the D/U ratio is also at best a vulnerable indicator of user preference. To infer user preferences from activities, one needs to dig deeper.

Although marketplaces have explored simple activity indicators such as the number of downloads or the number of users to complement user ratings of apps, unfortunately they are at best weakly correlated with app quality. This sounds discouraging, but can we find better indicators from app management activities?

4.2.2. Sequential Indicators. An important understanding of user activities is that they are not independent but always appear as sequences of events. Indeed, when search engines utilize users activities as implicit feedback about the relevance of documents, the sequences of actions are usually more indicative than single clicks. For example, Joachims et al. [2005] found that the sequence of actions where a user skips a higher-ranked search result but clicks on one ranked below emits a signal that the user finds the lower-ranked document more relevant than the higher-ranked one. Analogically, we anticipate that some sequential patterns of the app management activities may be better indicators of app quality than downloading actions alone. For example, we find that the sequence of uninstallation-downloading (UD) activities (i.e., a user uninstalls an app and later on reinstalls it) may be a good indicator of the user liking the app (which is why he reinstalled it). To verify this finding, we plot the proportion of these UD patterns among all users of an app and correlate it with the likerate of the app. Since not all apps have a UD activity sequence, we plot only those that have such a sequence.

Interestingly, the average number of UD sequences per user is generally positively correlated with the likerate of the app (Figure 10). This is promising, indicating that some user activity sequences may be good indicators of user preferences and app quality. Unfortunately, the UD sequences are relatively rare. Among 30,614,327 user activity sequences that have at least two actions, only 4% contain a subsequence of UD. If a marketplace uses this metric directly, it may yield even smaller coverage than the online ratings. Are there other activity patterns that are both indicative and more

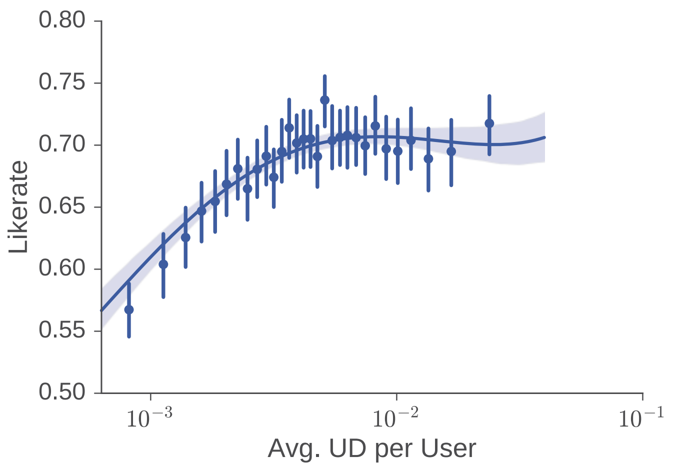


Fig. 10. Binned plot of the correlation between the likerate and *UD*.

frequent? Can these patterns, although individually weaker signals of app quality, be combined into stronger indicators? Although generally positive, the correlation in Figure 10 is still quite weak. How much better can sequential indicators from app management activities assess app quality? We are motivated to find all such sequential indicators and combine them in a systematic way.

5. A COMPARATIVE ANALYSIS

The correlation between the *UD* sequence and the likerate seems promising, which inspires us to find all sequential indicators like those and study them in a systematic way. We focus on answering an important question: how much better can these indicators measure user preferences and app quality compared to the existing measures used by the app marketplaces? The answer to this question may motivate the marketplaces to adopt these indicators and improve their current systems. To find this answer, we need a formal task and evaluation metric(s) with which any combination of activity indicators can be fairly compared to the existing indicators currently used by the app marketplaces.

5.1. The Ranking Task

To help users find and explore high-quality apps, most marketplaces provide rankings of apps based on either user ratings or other indicators. Ranking by the number of downloads or the number of users is particularly useful when the ratings are rare. To fairly compare new and existing indicators, we adopt this practical scenario and rank the same set of apps based on one indicator or a set of indicators combined through an algorithm. A method is better if it ranks high-quality apps higher and low-quality apps lower.

However, obtaining the ground truth of app quality is difficult, if there exists one at all. In this analysis, we use the ratings of apps as a surrogate of app quality. More specifically, we use the likerate of an app as the standard of its quality, as defined in Equation (2). Indeed, user ratings suffer from sparseness and biases, and cannot be aligned with the version of an app. However, it makes sense to judge the app quality/popularity when the user ratings are plentiful. In addition, we have to rely on them when nothing better is available. Readers could reproduce the results when a more reliable gold standard becomes available.

To ensure that the metric of likerates is reliable, we only look at apps with sufficient ratings. How many is sufficient? As in Section 4.2, we could use a threshold cutoff such as five ratings. In this analysis, we use a more rigorous definition, which will be described in Section 5.2. Although in the evaluation process we look at only apps with sufficient ratings, it should be noted that the activity indicators assessed to be effective can be easily applied to unrated or rarely rated apps and can provide a meaningful ranking of those.

5.2. Evaluation Metrics

Kendall's tau. When using ratings as the gold standard, the goodness of a ranking method (either by a single indicator or by a combination of indicators) can be measured by how much the ranking of apps it produces agrees with the ranking of apps by the likerates. Statistically, this agreement can be measured by Kendall's tau, which is widely used to compare two ranking lists [Manning et al. 2008]. It measures the difference between the number of concordant pairs in two ranked lists and the number of discordant pairs, normalized by the total number of ordered pairs:

$$\tau = \frac{\sum_{i,j}(\mathbb{I}[(x_i - x_j)\cdot(y_i - y_j) > 0] - \mathbb{I}[(x_i - x_j)\cdot(y_i - y_j) < 0])}{\sum_{i,j}(\mathbb{I}[(x_i - x_j)\cdot(y_i - y_j) > 0] + \mathbb{I}[(x_i - x_j)\cdot(y_i - y_j) < 0])}, \tag{3}$$

where i, j are two apps, x_i, y_i are the orders of the two apps in two ranking lists, and $\mathbb{I}[\cdot]$ is an indicator function that values 1 if the expression is true and 0 otherwise. The value of τ approaches 1 if the two rankings match perfectly, –1 if they are completely opposite, and 0 if they are independent.

Although Kendall's tau has a solid statistical foundation, there are two limitations when applying it to practice. It takes the whole list into consideration where top-ranked items contribute equally to items ranked at the bottom. In reality, the users normally browse apps from the top, and therefore having high-quality apps ranked to the top is more important than having low-quality apps ranked at the bottom. Moreover, like other correlation scores, τ is hard to be interpreted intuitively, which is a critical issue for the marketplaces.

Mean average precision. To complement Kendall's tau, we introduce another evaluation metric, the mean average precision (MAP), which is widely used in information retrieval literature to measure the relevance of a ranked list of documents [Manning et al. 2008]. The advantage of MAP is that it puts more weight on the top-ranked items, and as a "precision" its value is somewhat interpretable. However, MAP takes a binary value (good or bad) as the ground truth of every item (unlike Kendall's tau, which takes a ranking of items as the truth). In our data, such a binary decision is not directly available, which must be transformed from the likerates.

A straightforward way of doing this is to treat all likerates that are greater than 0.5 as "good" and those less than 0.5 as "bad." This is equivalent to summarizing a voting system with majority votes, a commonly adopted strategy in election and crowd-sourcing [Easley and Kleinberg 2010]. When the votes are rare, this simple strategy becomes problematic. Even when an app has abundant ratings, one single rating could have flipped the sign of the gold standard. Therefore, we filter the apps using a statistical test and only trust those with a likerate significantly higher or lower than 0.5 (there are significantly more likes or dislikes in the ratings). We used a two-tail proportion test with a confidence level of 0.05. Apps that pass this test would have many ratings and a considerable difference between positive and negative ratings. To ensure that the comparisons are fair and the results are faithfully compliant to the true reality, in the rest of this article, all methods will be evaluated using the apps in the test set that has passed the proportion test. We present the details of the experimental setup later in Section 5.4.1.

5.3. Baseline: Ranking by Popularity

As described, most marketplaces are already using popularity indicators, such as the number of downloads or the number of users to rank and recommend apps. How effective are these indicators?

We use Kendall's tau to measure the effectiveness of the rankings of apps by the two popularity indicators. Ranking by the number of users yields a τ of –0.244, and ranking by the number of downloads yields a τ of –0.237 on the test set of 1,423 apps. This is disappointing, showing that the current indicators used by the marketplaces do not accurately reflect the app quality, even negatively correlated. It is surprising, however, as the perceived quality of popularity ranking does not seem that bad. One possibility is that although the complete ranking is ineffective, it is probably doing a good job on top-ranked apps (and the users always browse from the top). Indeed, the MAP scores of ranking using the number of users and the number of downloads are 0.857 and 0.863, which look reasonable.

This contradiction between two metrics is actually meaningful: apps with many good ratings are more likely to be used and therefore rise to the top of the popularity list—yet another process of "rich-get-richer." A reasonable MAP and a miserable τ presents good news to the popular apps and bad news to the long-tail and new apps.

Having the understanding of the baselines, we are eager to know how much the sequential indicators from app management activities could improve the ranking of apps. Will they improve the ranking overall, thus providing an opportunity to lesser-rated apps? If so, will the improvement be at an expense of compromising the top-ranked apps? Can multiple indicators be combined as a joint force? In the following, we present a comprehensive evaluation of the predictive power of activity indicators.

5.4. Predictive Power of Activity Indicators

We believe that there are multiple patterns of app management activities that are good indicators of user preferences and app quality in addition to the *UD* pattern studied in Section 4. Although one can test the effectiveness of such patterns one by one, a more interesting question is whether the combination of these patterns can rank the apps much better. There could be many ways to combine multiple indicators. We cast this as a machine learning problem. In other words, we treat each potential activity indicator as a feature and learn a model from data that combines these features and makes predictions of the quality of an app.

5.4.1. Experimental Setup. Specifically, given any set of activity indicators (features), we train a regression model that predicts the likerate of an app. We explore various standard and state-of-the-art machine learning algorithms for this purpose, including ridge regression (Ridge) [Hoerl and Kennard 1970], lasso regression (Lasso) [Tibshirani 1996], random forest (RF) regression [Breiman 2001], and gradient boosted regression tree (GBRT) [Friedman 2002]. These algorithms provide a representative coverage of methods that combine features linearly and nonlinearly.[12]

To evaluate the predictors, from the set of 57,016 apps that have at least one rating and at least one nontrivial management activity sequence on Wandoujia (see Figure 4), we randomly select 50,000 apps for training and hold out the remaining 7,016 apps as the test set (the same split in Section 5.2 and 1,423 after filtering by the proportion

[12]We use the glmnet package [Friedman et al. 2010] in R for Lasso and Ridge and the scikit-learn package in Python [Pedregosa et al. 2011] for RF and GBRT.

Table II. Performance of Unit Features (Five Features)

Model	Metrics	Parameter Tuning: Tau	Parameter Tuning: MAP
Ridge	Tau	.044	.044
	MAP	.927	.927
Lasso	Tau	.044	.044
	MAP	.927	.927
RF	Tau	.057	.057
	MAP	.929	.930
GBRT	Tau	**.092**	**.083**
	MAP	**.933**	**.931**

Note: The parameter is selected using either tau or MAP. The best performance under two metrics are highlighted.

test). All prediction methods in this article are evaluated, and all evaluation metrics are computed using these 1,423 apps. The hyperparameters of every algorithm are selected using a fivefold cross validation on the training set (by optimizing either Kendall's tau or MAP), including the regularization parameter for Ridge and Lasso and the number of trees and number of nodes for RF and GBRT. We then train a model with the best-performing parameters on the entire training set and evaluate it on the test set.

5.4.2. Feature Extraction. We start with a set of simple indicators and then move to two other sets that make use of the sequential patterns and the time interval information. In the rest of this article, we use D to represent download, P for update, and U for uninstallation actions.

Unit features. We first extract a set of features for every app that includes the number of devices (denoted as #Dev), the average number of actions per device (denoted as Avg.Act), and the number of unit activities ($D/P/U$) per device. These simple features can be easily adopted by the production systems of marketplaces. Note that some of the features are already considered by the marketplaces (i.e., #Dev and D).

The performance of models with unit features is presented in Table II. Compared to the ranking methods currently used by the marketplaces, even just adding other unit features will largely improve performance. The improvements of MAP from 0.857 to 0.863 to 0.933 and tau from –0.244 to –0.237 to 0.092 over the two baselines are statistically significant with a p-value $\ll 0.01$. The significance level of MAP improvement is tested with a two-sided paired t-test and tau improvement with a randomization test [Smucker et al. 2007]. The same tests are used for the rest of the article. This improvement appears not only in the entire ranking list (Kendall's tau becomes positive) but also among the top-ranked apps (a significant improvement of MAP). This is promising, indicating that a combination of activity indicators better predicts the quality of both popular apps and the long tail. Among the four algorithms, GBRT achieves better results under both metrics, and tree-based methods outperform the linear regressions overall.

*Sequential features (sequence-*n*).* The success of unit features motivates us to further explore more complicated activity indicators—patterns like *UD* studied in Section 4. Compared to unit indicators that treat management activities independent of each other, the order between consecutive actions are likely to be a good indicator of user preference. For example, the behavior of downloading an app and then uninstalling it

Table III. Performance of Sequential Features (20, 68, 212, and 644 Features, Respectively) with Model Parameters Selected Using Tau and MAP (Columns Labeled)

		Feature Set and Parameter Tuning							
		Sequence-2		Sequence-3		Sequence-4		Sequence-5	
Model	Metric	Tau	Map	Tau	Map	Tau	Map	Tau	Map
Ridge	Tau	.074	.074	.078	.078	**.080**	.079	.077	.076
	MAP	.921	.921	.923	.923	.924	**.924**	.922	.923
Lasso	Tau	.100	.074	**.103**	.078	.103	.080	.103	.078
	MAP	.905	.921	.903	.923	.903	**.924**	.903	.920
RF	Tau	.067	.068	.069	**.071**	.071	.069	.070	.070
	MAP	.933	.934	.936	**.938**	.937	.936	.937	.937
GBRT	Tau	.117	.115	**.118**	.110	.109	.109	.107	.107
	MAP	**.942**	.942	.941	.941	.939	.939	.940	.940

Note: For every algorithm and every metric, the best-performing sequence-n features are shown in bold.

sends an opposite message to the behavior of uninstalling an app and then downloading it again. In general, if we consider the actions of every user managing every app as an ordered sequence, then such ordered activities appear to be a subsequence of length two.

To systematically study these sequential indicators, for every user of every app that has at least two actions, we construct a sequence of five possible symbols: three actions (D, P, and U) and two auxiliary symbols (start (S) and end (E)), which indicate the start and end of a sequence. For example, the sequence $SDPPUDE$ indicates that the user first downloaded the app, updated it twice, uninstalled it, and then decided to reinstall it, with no further actions recorded. The unit indicators ($D/P/U$) can now be seen as the subsequences of length one (or unigrams) and the UD and DU patterns can be seen as bigrams. Given any n, we can enumerate all subsequences with n or fewer *consecutive* actions. The average number of these n-grams per device, together with #Dev and Avg.Act, constitute the set of sequence-n features. For example, the sequence-2 feature set includes #Dev and Avg.Act, 3 unigram features ($D/P/U$), and 15 bigram features ($SD/SP/SU/DD/DP/DU/DE/PD/PP/PU/PE/UD/UP/UU/UE$). Notice that some sequential features, like UP (uninstallation followed by updating), might sound counterintuitive, as in reality one cannot update an app without downloading it first. This is because only activities that are conducted through the Wandoujia app are logged, and the user may have downloaded the app from other sources. In addition, a sequence does not necessarily begin with D, as the app may be preloaded or installed before the 5-month period. We vary the n from one to five in our experiments, as a larger n may make some features very sparse and lead to overfitting.

The performance of sequential features is presented in Table III. Note that sequence-(n-1) features are always a subset of sequence-n features. Most of the best results are observed when length-3 or length-4 features are used. When the length of subsequences is greater than four, the performance begins to drop. Compared to unit features, the best-performing sequential features increases tau from 0.092 to 0.118 (p-value = 0.16) and MAP from 0.933 to 0.942 (p-value $\ll$ 0.01).

With an investigation of the importance of individual features (e.g., the coefficients in Ridge), we notice that there are quite a few incomplete patterns, such as UU and UP. Some actions must have been omitted in these patterns simply because they were not captured by the Wandoujia app. App management activities only represent a small subset of activities that a user does with an app. Between consecutive $D/P/U$ activities, there may be other actions (e.g., clicking, browsing, and grouping) that are not logged but quite indicative. This suggests that the predictive power of the activity patterns may still be underestimated. Without observing these activities, we are not able to verify the hypothesis. Nevertheless, the time interval between two consecutive

Table IV. Performance of T-Sequential Features with Model Parameters Selected Using Tau and MAP (Columns Labeled)

		Parameter Tuning			
		Tau		MAP	
Model	Metric	w/o FS	w/ FS	w/o FS	w/ FS
Ridge	Tau	.094	.103	.094	.103
	MAP	.929	.935	.930	.935
Lasso	Tau	**.172**	.172	.095	.103
	MAP	.921	.921	.928	.935
RF	Tau	.080	.075	.080	.074
	MAP	.944	.943	.944	.942
GBRT	Tau	.140	.136	**.140**	.136
	MAP	**.951**	.950	**.951**	.950

Note: Time intervals are inserted into sequence-3 features: 824 features before feature selection (w/o FS) and 153 features after (w/ FS).

app management activities may be informative even if we do not know what actually happened during the interval.

Time intervals (T-sequential). The length of time intervals between consecutive activities can be indicative. Intuitively, an uninstallation immediately following a download indicates a stronger negative preference than an uninstallation that happens weeks or months after downloading. Incorporating time intervals into the sequential activity patterns is not easy, as the length of the intervals is continuous. However, recall from Figure 3 that the intervals between consecutive activities follow a cyclic pattern every 24 hours and an exponential trend overall. Therefore, we decide to discretize the intervals into days. In other words, we insert a time symbol T into the activity sequences, with each of its appearances indicating a complete day (24 hours) between two consecutive activities. Two consecutive activities within a day will include a hyphen (-) in the new sequence. We also consider an interval longer than 4 days to be long enough and substitute four or more consecutive Ts with an asterisk (*). An activity sequence therefore becomes something like *SDTTP*P-UTDE*. We then fill in the sequential features with time intervals. For example, the feature *DPU* in the previous feature set is extended to *DTP-U*, *DTTP-U*, *D*PTTU*, and so on. These extended sequential features, together with #Dev and Avg.Act, constitute the set of time-interval–inserted sequential (T-sequential) features.

The consideration of time intervals has significantly increased the number of features, exposing the models to the potential threat of overfitting. We consider a feature selection strategy that filters out features not significantly correlated with the likerates. Specifically, we compute the Pearson correlation between every feature and the likerates of apps based on the training data, and only keep those with a p-value smaller or equal to 0.05. The performance of T-sequential features are presented in Table IV, with and without feature selection. We see that by inserting time intervals into sequential features, the best result of MAP increased from 0.942 to 0.951 (p-value $\ll 0.01$) and the best result of tau increased from 0.118 to 0.172 (p-value $\ll 0.01$), surprisingly achieved by Lasso. Feature selection generally has improved the linear methods but does not improve the tree-based methods, suggesting that the tree-based methods are more robust to overfitting.

Putting it all together. The last exploration we do is put all features together and test whether it further improves the ranking performance. As unit features are a subset of sequential features, we essentially combine sequence-3 features and their time interval–inserted extensions. The same feature selection is used to reduce the

Table V. Performance of the Combination of T-Sequential Features and Sequential Features: 890 Features Before Feature Selection and 207 Features After

		Parameter Tuning	
Model	Metric	Tau	MAP
Ridge	Tau	0.103	0.103
	MAP	0.935	0.935
Lasso	Tau	**0.172**	0.102
	MAP	0.921	0.934
RF	Tau	0.077	0.077
	MAP	0.943	0.943
GBRT	Tau	0.140	**0.141**
	MAP	**0.954**	**0.952**

number of features. The results are shown in Table V. We see that GBRT still achieves the highest MAP, bringing the metric to above 0.95. This MAP improvement over the best result of T-sequential features is significant with a p-value $\ll 0.01$. The best result of tau does not improve.

5.5. Discussion

Overall, combining multiple indicators extracted from app management activities significantly outperforms the ranking methods used by the marketplaces, increasing Kendall's tau from –0.24 to 0.17 and the MAP from 0.86 to 0.95. The result is encouraging, showing that these activity indicators improve the accuracy of both the complete ranking of apps (good news to unpopular and new apps) and the top-ranked apps (good news to popular apps). Indeed, during the experiments, we observed some rather interesting behaviors of the predictors, which may provide insights for the marketplaces on how to better rank the apps and how to explain the ranking.

5.5.1. Feature Analysis. One surprising observation is that the highest tau score (0.1716) is achieved by Lasso (when the hyperparameter is tuned using tau). In most other cases, the performance of Lasso is inferior to GBRT and RF. We found that with Lasso, only one feature has a nonzero coefficient, which is the pattern *SD-U* from the T-sequential feature set. This means that the first action that we observed from a user is downloading the app, and then he uninstalled the app within 24 hours. Intuitively, this suggests that the user downloaded the app, tried it a few times, found it disappointing or even annoying, and then uninstalled it without hesitation. The time interval is quite sensitive. Lasso also identified the corresponding *SDU* from the sequential patterns, but its predictive power is much weaker (0.10 vs. 0.17). The coefficient of the feature is negative, meaning that the smaller fraction of users who did this, the higher the quality of the app. This is rather interesting, suggesting that if we care about the accuracy of the orders between all apps, especially the apps that are not ranked at the top (unpopular/new apps or low-quality apps), one indicator is better than many. A marketplace could immediately adopt this finding in its system and provide a ranking of apps in the reverse order of this indicator, with an easy explanation of the top-ranked apps: “everyone who installs it keeps it.” We anticipate that this would be particularly useful for promoting new and high-quality apps.

It is much harder if our goal is to optimize the top-ranked apps. To achieve a high MAP, every method has utilized a combination of many features. We investigate the best performer, achieved by GBRT with both the T-sequential features and the sequence-n features. In particular, we look at the importance of each feature in the GBRT model and the Pearson r correlation between the feature and likerate. We observe that many

Table VI. Summary Statistics of the Most Over/Underpredicted Apps—Apps with Prediction Errors Significantly Different from the Average

	All	Overpredicted	Underpredicted
Apps (#)	1423	162	9
Avg. Users (#)	2379.18	343.55	80.44
Avg. Ratings (#)	366.64	68.95	407.33
Games (%)	36.61	45.06**	0.00
News Readers (%)	5.56	0.62**	11.11
Avg. Abs. Error	0.242	0.482	0.576

Note: **indicates significance at 0.05 level. Only the Game and New Readers categories have significantly different proportions of apps.

of the important features are variants of *DU*, such as *SD-U*, *D-UE*, and *D-U*, and they all have negative correlations with the likerate ($r < -0.18$). Interestingly, *D*UE* is also among the most important features but is positively correlated with the likerate ($r = 0.02$). Notice that *D*UE* requires a long interval (≥72 hours), so the users must have kept the app for some time. Other important features include the variants of *DD* and the variants of *UU*. The *UD* indicator that we presented in Section 4 is ranked 48th by importance in the best-performing GBRT among the 207 features in total.

5.5.2. Error Analysis. Both metrics show an overall agreement between the predicted result and the observed likerate. There are still many mistakes made in the predictions. Admittedly, all algorithms have their limits. Yet we are curious about whether the "gold standard" is well grounded. In other words, do the observed likerates reflect the true preference of the users? Table VI shows details about the most overpredicted and underpredicted apps—apps with a predicted likerate much higher or lower than observed. Compared to the statistics of all test apps, both overpredicted and underpredicted apps have fewer ratings and fewer users, implying the curse of data sparseness. Among the most underpredicted apps, there are generally more ratings than users. Although the popularity of apps might rise and fall, and the 5-month data might not include all users, the unexpected high ratio of ratings over users still makes us suspicious of rating manipulations.

Among the 1,423 apps in the test set, 36.6% of them are games, constituting the largest category. However, the ratio of games is 45.1% among the significantly overpredicted apps, which differs significantly from the overall ratio (p-value $= 0.04$, χ^2 test). One possible explanation is that users might have different expectations and attitudes to different categories of apps. Another possibility is that game users are less likely to leave positive ratings and more likely to leave negative ratings, probably due to the difference in user demographics. Similarly, we see that the proportion of News Readers is smaller in the overpredicted apps (p-value $= 0.01$, χ^2 test). In either case, this suggests biases in online ratings, which limit their effectiveness as the gold standard of app quality. This also suggests that incorporating additional information (e.g., app category) to user behaviors may further improve the performance of prediction. In the preceding analysis, we choose not to use information other than app management activities, as the goal is to fairly measure the effectiveness of the activity indicators. Additional information such as app profiles, user demographics, and textual reviews may be explored in the future task to optimize the prediction of app ratings.

6. CORRECTING SELECTION BIAS THROUGH IPW

The error analysis presented in Table VI motivates us to further consider whether systematic biases exist in the analysis pipeline and whether correcting these biases could further improve the prediction performance.

In our experimental setup, we randomly split apps into a training set and a test set. The training and tuning process is performed on the entire training set. Ideally, we could use all apps in the testing set to evaluate the trained models. However, not all apps have enough ratings to determine whether users' attitudes toward them are positive or negative. In fact, for many apps that received only a few ratings, any additional rating, either *like* or *dislike*, could significantly diverge the average likerates. To ensure that the gold standard for evaluation is robust, we had to filter the apps in the test set and keep only the apps that have significantly more likes or dislikes (see Section 5.2 for details). Indeed, only 20.2% apps are left after the proportion test.

The proportion test, although a principled treatment, creates biases between the training set and the filtered test set. Features of the apps in the filtered test set no longer share the same distribution as those of the apps in the training set. Intuitively, the more people using an app (represented by the *NSEQ* feature), the more ratings it may get, and the more likely that it passes the proportion test. The filtered test set, therefore, may not be the perfect set to evaluate the models learned from the training set. In addition, the apps in the training set that would have been filtered out (those with noisy likerates) may affect the performance of the prediction models.

An easy way out is to apply the same filter to the training set as well. By doing this, we can artificially match the distributions in training and test and potentially improve the prediction performance. However, we will also lose information by throwing away 80% of the apps in the training set. These apps are not randomly picked and are more likely to be newly released or unpopular ones. Such filter can result in a mismatch between the distribution of the selected apps for analysis and the distribution of the true population of apps, thus putting apps with fewer ratings in an unfair position in our ranking task.

Essentially, the biases result from nonrandomly selecting examples for testing proposes. Such biases are also known as selection biases, which have been well studied in other domains, such as statistics [Little and Rubin 2014], economics [Heckman 1979; Imbens and Wooldridge 2009] and epidemiology [Hernán et al. 2004], yet they have been largely neglected in data mining and machine learning studies (except for very few studies, such as Huang et al. [2006]).

Correcting selection biases is critical for drawing convincing conclusions from behavioral analyses. To this end, there have been various methods proposed in the statistics and econometrics literature, such as instrumental variables, regression discontinuity, matching, and weighting methods [Imbens and Wooldridge 2009; Blundell and Dias 2009; Imbens and Rubin 2015]. Different approaches require different assumptions or certain types of features being available. In this work, we adopt a principled statistical method, namely IPW [Rosenbaum and Rubin 1983], which is commonly used to deal with selection biases in observational studies. Compared to other methods, IPW is the most applicable approach for our setting, and its assumptions are satisfied in our data. Two recent studies [Schnabel et al. 2016; Wang et al. 2016] also adopted this method in different machine learning tasks. These studies were published while this article was under review.

Unless specified, in the remainder of this section we will use the term *selection* or *selected* to denote an app passing the proportional test and having significantly more likes or dislikes.

6.1. Inverse Probability Weighting

Let $X \in R^n$ denote the feature representation of an app and $y \in [0, 1]$ denote the likerate of an app. We use a binary variable $d \in \{0, 1\}$ to represent whether the app is selected. $d = 1$ means that the app is selected—that is, it passes the proportional test. Our goal is to learn a model to predict y based on X while we only know the true y values for the selected apps. The other apps with "missing" y values cannot be used for training directly. Selection biases are introduced if we simply ignore these apps.

If we can assume that apps with similar properties (denoted as X) have similar ratings (denoted as y), denoted as $y \perp\!\!\!\perp d \mid X$, we could replace the missing y values with the y values of similar apps (where similarity has to be defined carefully). For each unselected app, we may match it with some selected apps based on its feature representation (matching on X) and assign the y values of the matched apps as the y values for this unselected app. In this way, all apps have y values and can be used for training, and the selection biases are corrected.

However, finding similar apps may be difficult, and we may easily fall into "the curse of dimensionality." Rosenbaum and Rubin [1983] prove that if $y \perp\!\!\!\perp d \mid X$, then $y \perp\!\!\!\perp d \mid X \Rightarrow y \perp\!\!\!\perp d \mid P(X)$, where $P(X) = Pr(d = 1|X)$. $P(X)$ is the probability of being selected, also known as the propensity score. Therefore, instead of finding apps with similar features, we can find apps with similar probabilities of being selected. In other words, we are matching on $P(X)$.

Considering that in the real world an app is either selected or unselected, we can only estimate the propensity score. Once the propensity score is estimated, we do not necessarily need to perform the matching. Instead, we can reweight each selected app with the inverse of its propensity score. Each selected app represents $1/P(X)$ apps that share the same propensity score. The less likely an app is selected (smaller $P(X)$), the more similar apps (in $P(X)$) it represents. After reweighting, the selected apps represent all training apps regardless of being selected or not, and they have no missing y value. To ensure that apps of all propensity scores are represented, there should be selected apps around the propensity score of each unselected app and vice versa. In other words, the support of the propensity scores of both selected and unselected apps should overlap. Such a requirement is known as the common support assumption, which can be verified once propensity scores are estimated.

Therefore, IPW follows a two-stage process: in the first stage, we estimate the propensity score. This is done by training a classifier that predicts whether or not an app is selected. The trained classifier is then applied to all apps to predict the probabilities of being selected. In the second stage, which is the analysis stage, we reweight the selected apps and build a regression model to predict their likerates, as shown in Section 5.4.

Intuitively, IPW can be applied to any data mining scenario where items with certain properties are overrepresented or underrepresented in the sample (compared to the true population). It works by strengthening the voice of underrepresented items and downweighting the voice of overrepresented items in the mining stage.

Before going through the details of the two stages, we introduce the experiment setting and the baseline for comparison. We choose to start with the model with the highest tau (0.172) in Section 4.2.2, namely Lasso on T-sequential features, with the parameter tuned to optimize tau. We also choose Lasso on sequence-2 features. This feature set includes only 20 features, making it easier to interpret the result.

We summarize the results of first filtering the training set and then correcting selection biases later in Table VIII, where the first row is copied directly from Tables III and IV for comparison. We use the same training/test set split and report the tau/MAP on the test set when the hyperparameter is tuned to optimize tau/MAP with fivefold cross validation on the training set. We also report the number of features selected when tau is optimized.

Because we filter both the training set and the test set, the data becomes sparser. To ensure that it does not result in awfully skewed feature distributions, we performed feature normalization for all apps (in training and test): since the number of users follows a power law distribution, we take the logarithm of the *NSEQ* feature; we also bound the value of every feature within three standard deviations from its mean,

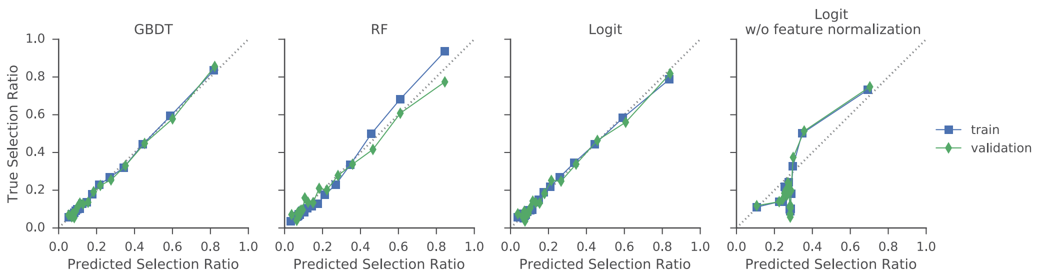


Fig. 11. Binned average propensity score and true selection ratio for each algorithm.

Table VII. First-Stage Performance Summarization

Model	Train Accuracy	Validation Accuracy
GBDT	.846	.838
RF	.865	.838
Logit	.839	.839

calculated based on the training set. As shown in the second row of Table VIII, this feature normalization alone barely affects the performance of the baseline.

Expectedly, filtering both the training and test set without reweighting leads to a sharp increase in tau, bringing it from 0.17 to 0.28 on T-sequential features and from 0.10 to 0.25 on sequence-2 features, as shown in the third row of Table VIII. Such a benefit comes from matching the distributions of the training set and the test set. As mentioned, however, such an improvement may be an artifact of the selection biases and may not generalize to reality. We need to correct the selection biases using IPW.

6.1.1. First Stage: Propensity Score Estimation. In the first stage, we estimate the propensity score of each app—that is, the probability of the app being selected. Because we observe the binary outcome, whether or not the app can pass the proportion test, the first stage can be accomplished by training a classification model. In our case, we compare various classification algorithms, including a linear method, logistic regression (Logit), and two nonlinear methods, gradient boosted decision tree (GBDT) and RF. We used sequence-2 features as introduced in Section 5.4.2, as adding more features may lead to overfitting for the first stage.

The classification models are both trained with and applied to apps in the training set. However, it is not enough for the classifier to achieve good accuracy. More importantly, the classifier should not overfit. Therefore, we use cross validation to find the best model: we randomly sample 20% of apps from the training set as a hold-out validation set and use the remaining 80% of the training set to train models. The training and validation accuracy for each algorithm is reported in Table VII. It can be observed that all three models achieve an accuracy of 83.8% to 83.9% on the validation set. RF reaches a higher training accuracy of 86.5%, which may suggest overfitting.

On average, only one out of five apps are selected, so guessing all apps as not selected would yield a baseline accuracy of 80.2%. Considering this, the 83% accuracy that we get is not astonishingly high. However, the purpose of the first stage is not to predict whether an app is selected but rather to establish a good estimate of the probability that an app with certain properties is selected. To evaluate the quality of the propensity score, we partition the apps into equal-size bins based on their propensity scores. Since we know whether an app is actually selected or not, we calculate the selection ratio of apps in each bin and plot it in Figure 11.

Table VIII. Effectiveness of IPW

Filtered Training	Normalized Features	IPW	T-Sequential			Sequence-2		
			Tau	MAP	Features (#)	Tau	MAP	Features (#)
No	No	—	.172	.921	1	.100	.921	1
No	Yes	—	.172	.939	1	.100	.924	1
Yes	Yes	—	.284	.940	51	.253	.926	17
Yes	Yes	GBDT	.320	.933	21	.306	.918	5
Yes	Yes	Logit	.322	.932	21	.304	.919	6

Note: All (second-stage) regressions use Lasso. The reported tau and MAP scores are based on models optimizing tau and MAP, respectively.

Table IX. Summary Statistics of the Most Overpredicted Apps with IPW Prediction

	All	Overpredicted
Apps (#)	1,423	183
Avg. Users (#)	2,379.18	162.77
Avg. Ratings (#)	366.64	42.60
Games (%)	36.61	47.00***
News Readers (%)	5.56	0.55***
Avg. Abs. Error	0.202	0.628

Note: ***indicates significance at 0.01 level. Apps whose prediction errors are significantly different from the average are identified as over/underpredicted. No underpredicted apps are identified. Only the Game and New Readers categories have significantly different proportions of apps.

If the propensity score is estimated correctly, the selection ratio of each bin should equal the mean propensity score of the apps in the bin. In other words, we should expect a line close to $y = x$. Indeed, we observe that for GBDT and Logit, the bin-bin curves for both the training and validation sets are quite close to the $y = x$ line, whereas for RF, the curves divert from $y = x$ and fall into different directions. For comparison, we also plot an ill-estimated model (Logit without feature normalization), where both curves are far away from $y = x$. We can conclude that GBDT and Logit both give us reasonable estimations of propensity scores. Indeed, the two methods produce similar propensity scores for most apps, with $\rho = 0.98$ and $RMSE = 0.035$. Figure 11 also indicates that there are indeed both selected and unselected apps in each propensity score bins, ensuring that the common support assumption in IPW is satisfied.

In the next stage, we will use two sets of propensity scores, one estimated by GBDT and the other by Logit, to reweight the training instances.

6.1.2. Second Stage: Regression Model with Reweighted Apps. We reweight the training instances with the inverse of the propensity scores estimated in the first stage and redo the likerate prediction task. All four preceding models can deal with weighted regression. Here, we choose Lasso to build all second-stage prediction models. With Lasso, we can identify the features selected in different settings (i.e., with/without filtering or IPW). The coefficients estimated for different features can also be examined, which provides an interpretable way to understand the changes caused by IPW.

When comparing the last three rows in Table VIII, we see that after reweighting, the best tau score increased significantly from 0.284 to 0.322, whereas the best MAP score decreased slightly. The results are consistent whether GBDT or Logit is used in the first stage of IPW, demonstrating the robustness of the process. Even with sequence-2 features, we are able to reach a tau score above 0.30 (compared to the best tau score 0.172 in Section 4.2.2). The large improvement in tau and minimal influence on MAP is intriguing. Remember that tau measures the ranking accuracy of the entire ranked list, whereas MAP highly focuses on top-ranked items—this suggests that IPW effectively improves the predictions for the majority of apps but does not necessarily improve the predictions for top-ranked apps. This reassures the effectiveness of IPW as the reweighting benefits underrepresented apps, in our case mostly the long-tail apps. The popular apps have a high probability to be selected anyway and thus are less likely to be affected by IPW.

The purpose of correcting selection biases is not only to improve the prediction accuracy on the test data but also to correct the learned model so that it more generally reflects the reality. It is interesting to validate this by looking at the features selected by Lasso before and after IPW (when the hyperparameters are tuned to optimize tau).

As mentioned in Section 4.2.2, before the training set is filtered, only one feature is selected from either feature set (*DU* from the sequence-2 set and *SD-U* from the T-sequential set). The trained models inevitably underfit. When the training set is filtered but not reweighted (where selection biases are introduced), many more features are selected (51 from the T-sequential set and 17 from the sequence-2 set). With IPW, however, the number of selected features drops significantly (21 from the T-sequential set and 5 from the sequence-2 set). IPW improves the generalizability of the trained models, where a model with a much smaller degree of freedom can achieve a higher test accuracy.

We can further compare the coefficients of particular features across different settings. One extremely interesting feature is *DU* in the sequence-2 set. Before the training set is filtered, the coefficient of *DU* is –0.01, which can be naturally explained, as uninstalling an app right after downloading implies negative preference. When the training set is filtered but not yet reweighted, the coefficient becomes 0.05, with the sign flipped. After IPW, the sign becomes normal again and the coefficients are -0.06 and -0.07 (reweighting based on GBDT and Logit, respectively). The flipping sign strongly suggests that simply filtering the training set introduces selection bias, which invalidates the model and makes it hard to interpret; such biases are corrected by IPW.

The other four features (and signs of their coefficients) selected from the sequence-2 set include *NSEQ* (–), *AVE_ACTION* (+), *DP* (+), and *PE* (+). The features selected from the T-sequential set also include *NSEQ* (–), *AVE_ACTION* (+), and multiple variants of DP, such as *SD*P* (+) and *D*PE* (+), and multiple variants of *DU*, such as *DUE* (–) and *SDTTTU* (–). Again, the prediction models trained after IPW are consistent and interpretable.

6.1.3. Error Analysis. Similar to Section 5.4.2, we analyze the prediction errors after applying IPW. Compared to the results before applying IPW (Table VI), no significantly underpredicted apps are found, but there are more overpredicted apps (i.e., 183 (after IPW) versus 162 (before IPW)). A closer look at these overpredicted apps reveals that they have even fewer users (162.77 vs. 343.55) and ratings (42.60 vs. 68.95) on average. On the one hand, with fewer users, the extracted features could be noisy, which can affect the prediction accuracy. On the other hand, with fewer ratings, the ground truth (average likerates) could have larger variation, which can also result in a large prediction error.

Games and news readers are still more likely to be overpredicted, which implies that the particular process of IPW can still be improved. For example, categories and profile information of apps may be considered at both stages, and it may be meaningful to separate the prediction of the number of ratings and the likerates. In general, one may apply more powerful machine learning tools to further improve the accuracy of propensity score estimation. We leave these potential improvements as future work.

7. IMPLICATIONS AND LIMITATIONS

Thus far, we have presented how app management activities can help predict app quality. In this section, we discuss some takeaway implications and potential limitations.

7.1. Implications

Many findings from our analysis are directly useful for app marketplaces, app developers, and data miners. We found that the number of downloads of an app is not a good indicator of users' preference or app quality; it actually yields a negative coefficient in the regression analysis. Using the number of downloads alone to rank apps may be misleading and simply triggers the bandwagon effect, which is unfair to new apps and high-quality niche apps. App marketplaces can immediately take actions. When ratings are rare, simple patterns of app management activities are much more robust indicators of user preference. Even using one single pattern, DU, the quality of apps can be ranked accurately and fairly, which is especially valuable for new and high-quality apps. Multiple time-aware sequential patterns can be combined with a machine learning algorithm to further improve the ranking accuracy.

For app developers, it is also important to understand that app management activities indicate user preferences. If possible, capturing early signals such as DU may help the developers identify potential problems of their apps early on and make improvements in time. Understanding the temporal patterns of app management activities can also help the developers effectively schedule the release and updates of their apps.

For data miners, a desirable takeaway is the effectiveness of IPW. Indeed, in many scenarios they had to prune the data for modeling purposes (e.g., dropping inactive users, infrequent words, short documents, long-tail queries). Such heuristic pruning improves model performance empirically but also introduces mismatched distributions and selection biases. IPW is a principled and powerful tool to correct selection biases, which may improve both the prediction performance and model consistency.

7.2. Limitations

Considerable care and attention have been given to ensure the rigor of our study. We are also aware of a few limitations that are not able to be conquered with our current analysis. Understanding them may help readers use our findings with caution.

First, this work relies on the app management activities to derive user preferences. These indicators are more ubiquitous and reliable, as user ratings may be sparse and of low quality. Combined with machine learning techniques, it is demonstrated that the app management activities can accurately predict user preferences. In reality, a model will be trained on apps with sufficient ratings and applied to apps with fewer ratings, as long as sufficient management activities are logged. The model could fail if an app has too few management activities. Still, there are way more apps with sufficient activities than apps with sufficient ratings. Those rarely rated apps with enough management activities would benefit tremendously from the prediction.

Second, the data we study are collected from a single app store. Although we demonstrated that ratings for the same apps mostly correlate over multiple app stores (see Section 4.1), there are considerable biases introduced by languages and cultures, which may result in different distributions of the types of apps, different types of devices, and even different distribution of ratings. Chen et al. [2013] compared the ratings of 1,464 equivalent apps between Apple App Store and Google Play and found that about 20% were not rated equally. Additionally, different app stores have different rating models for apps. For instance, Google Play employs the five-scale model, whereas Wandoujia employs binary votes. We believe that users behaviors of managing apps are more robust signals of user preference, but the market biases should still be carefully considered if cross-store data are available.

Third, our study does not distinguish the versions of apps. Since the dataset spans 5 recent months, it is quite possible that some apps have evolved over multiple versions. Users may like a specific version but dislike and uninstall another version of the same app. Unfortunately, Wandoujia did not distinguish the ratings committed to specific versions of apps, which may affect the robustness of our results. In other words, there may be mismatched distributions of apps and app versions between the behavioral data and the rating data. Our analysis may be biased toward newer apps and newer versions of apps. Such biases should be addressed when detailed metadata of app ratings are available.

Last but not the least, our study does not make further analysis on features other than management activities. For example, it is reported that there can be some device-specific bugs [Wei et al. 2016; Lu et al. 2016]. For instance, some apps may be incompatible or even crash down on specific device models or operation systems. One may also associate the app management activities with the brands and operating system information of devices, and it would be interesting to explore whether apps are more likely to be uninstalled under a particular environment. Additionally, the category information of an app can also be utilized in the machine learning models to further improve prediction accuracy. These signals are left out in the present analysis to keep our main conclusion clean.

8. CONCLUSION

In this article, we have presented an empirical analysis of a very large collection of app management activities of smartphone users. The data were collected through a leading Android marketplace in China. We demonstrated that users download, update, and uninstall apps differently when they like or dislike the apps. These app management activities indicate that the users "vote with their feet," which can effectively supplement the biases and sparsity of online ratings of apps. We identified behavioral patterns that serve as indicators of the user's preferences on apps, which can be integrated by machine learning algorithms that predict the ratio of positive ratings of the apps. With potential selection biases corrected by a principled statistical method, IPW, the prediction can be improved significantly.

Some surprising findings from our analysis may be directly useful for app marketplaces or app developers. For example, we demonstrated that the number of downloads of an app is not a good indicator of users' preference or the quality of the app. We also notice that users have a routine schedule to manage apps on their mobile devices. Several simple patterns of the activities provide a general ranking of apps that is surprisingly accurate, which may be used to effectively promote new and high-quality apps. Multiple time-aware sequential patterns can be combined with a machine learning algorithm and significantly improve the accuracy of top-ranked apps.

Since Wandoujia is a marketplace that is most popular in China, it is interesting to explore the uniquely local characteristics of Wandoujia and compare them to other marketplaces, such as Google Play. It is also a natural extension of this work to correlate the app management activities with the patterns of how users interact with the apps, or with the narrative reviews of apps. It is desirable to integrate the knowledge derived from the management behaviors into the recommender systems of apps.

ACKNOWLEDGMENTS

We thank the anonymous reviewers of the conference version of this article. Their comments were very constructive; some actually inspired our extended explorations.

REFERENCES

Sharad Agarwal, Ratul Mahajan, Alice Zheng, and Victor Bahl. 2010. Diagnosing mobile applications in the wild. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. 22.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439, 509–512.

Richard Blundell and Monica Costa Dias. 2009. Alternative approaches to evaluation in empirical microeconomics. *Journal of Human Resources* 44, 3, 565–640.

Matthias Böhmer, Brent Hecht, Johannes Schöning, Antonio Krüger, and Gernot Bauer. 2011. Falling asleep with Angry Birds, Facebook and Kindle: A large scale study on mobile application usage. In *Proceedings of the 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 47–56.

Matthias Böhmer and Antonio Krüger. 2013. A study on icon arrangement by smartphone users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2137–2146.

L. Breiman. 2001. Random forests. *Machine Learning* 45, 1, 5–32.

Rishi Chandy and Haijie Gu. 2012. Identifying spam in the iOS App store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*. 56–59.

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 *Proceedings of the 36th International Conference on Software Engineering*. 767–778.

Ying Chen, Heng Xu, Yilu Zhou, and Sencun Zhu. 2013. Is this app safe for children? A comparison study of maturity ratings on Android and iOS applications. In *Proceedings of the 22nd International World Wide Web Conference*. 201–212.

Gokul Chittaranjan, Jan Blom, and Daniel Gatica-Perez. 2013. Mining large-scale smartphone data for personality studies. *Personal and Ubiquitous Computing* 17, 3, 433–450.

David Easley and Jon Kleinberg. 2010. *Network, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press.

Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. 2010. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. 281–287.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software* 33, 1, 1–22.

J. H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics and Data Analysis* 38, 4, 367–378.

Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. 2013. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1276–1284.

Michael Goul, Olivera Marjanovic, Susan Baxley, and Karen Vizecky. 2012. Managing the enterprise business intelligence app store: Sentiment analysis supported requirements engineering. In *Proceedings of the 45th Hawaii International Conference on System Science*. 4168–4177.

Mark Harman, Yue Jia, and Yuanyuan Zhang. 2012. App store mining and analysis: MSR for app stores. In *Proceedings of the 9th IEEE Working Conference of Mining Software Repositories*. 108–111.

J. J. Heckman. 1979. Sample selection bias as a specification error. *Econometrica* 47, 1, 153.

Miguel A. Hernán, Sonia Hernández-Díaz, and James M. Robins. 2004. A structural approach to selection bias. *Epidemiology* 15, 5, 615–625.

Arthur E. Hoerl and Robert W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12, 1, 55–67.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 8th IEEE International Conference on Data Mining.* 263–272.

Jiayuan Huang, Arthur Gretton, Karsten M. Borgwardt, Bernhard Schölkopf, and Alex J. Smola. 2006. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*. 601–608.

Claudia Iacob and Rachel Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. 41–44.

Guido W. Imbens and Donald B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences*. Cambridge University Press.

Guido W. Imbens and Jeffrey M. Wooldridge. 2009. Recent developments in the econometrics of program evaluation. *Journal of Economic Literature* 47, 1, 5–86.

Bernard J. Jansen. 2008. *Handbook of Research on Web Log Analysis*. IGI Global, Hershey, PA.

Daxin Jiang, Jian Pei, and Hang Li. 2013. Mining search and browse logs for Web search: A survey. *ACM Transactions on Intelligent Systems and Technology* 4, 4, 57–37.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Web Data Mining*. 219–230.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 154–161.

Jaeyeon Jung, Seungyeop Han, and David Wetherall. 2012. Short paper: Enhancing mobile application permissions with runtime feedback and constraints. In *Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*. 45–50.

Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E. Hassan. 2015. What do mobile app users complain about? *IEEE Software* 32, 3, 70–77.

Huoran Li, Wei Ai, Xuanzhe Liu, Jian Tang, Gang Huang, Feng Feng, and Qiaozhu Mei. 2016. Voting with their feet: Inferring user preferences from app management activities. In *Proceedings of the 25th International Conference on World Wide Web*. 1351–1362.

Huoran Li, Xuanzhe Liu, Wei Ai, Qiaozhu Mei, and Feng Feng. 2015a. A descriptive analysis of a large-scale collection of app management activities. In *Proceedings of the 24th International Conference on World Wide Web Companion*. 61–62.

Huoran Li, Xuan Lu, Xuanzhe Liu, Tao Xie, Kaigui Bian, Felix Xiaozhu Lin, Qiaozhu Mei, and Feng Feng. 2015b. Characterizing smartphone usage patterns from millions of Android users. In *Proceedings of the 2015 ACM Conference on Internet Measurement*. 459–472.

S. Lim, P. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden. 2015. Investigating country differences in mobile app user behavior and challenges for software engineering. *IEEE Transactions on Software Engineering* 41, 1, 40–64.

Roderick J. A. Little and Donald B. Rubin. 2014. *Statistical Analysis with Missing Data*. John Wiley & Sons.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies* 5, 1, 1–167.

Xuan Lu, Xuanzhe Liu, Huoran Li, Tao Xie, Qiaozhu Mei, Dan Hao, Gang Huang, and Feng Feng. 2016. PRADA: Prioritizing Android devices for apps by mining large-scale usage data. In *Proceedings of the 38th International Conference on Software Engineering*. 3–13.

Walid Maalej and Hadeer Nabil. 2015. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *Proceedings of the 23rd IEEE International Requirements Engineering Conference*. 116–125.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. 2016. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering* 21, 3, 1067–1106.

Mark E. J. Newman. 2005. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics* 46, 5, 323–351.

Myle Ott, Claire Cardie, and Jeff Hancock. 2012. Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st World Wide Web Conference*. 201–210.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 309–319.

Fabio Palomba, Mario Linares-Vásquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. 2015. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In *Proceedings of the 31st IEEE International Conference on Software Maintenance and Evolution*. 291–300.

Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the 31st IEEE International Conference on Software Maintenance and Evolution*. 281–290.

Dae Hoon Park, Mengwen Liu, Cheng-Xiang 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*. 533–542.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 10, 2825–2830.

Ahmad Rahmati, Chad Tossell, Clayton Shepard, Philip Kortum, and Lin Zhong. 2012. Exploring iPhone usage: The influence of socioeconomic differences on smartphone adoption, usage and usability. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 11–20.

Ahmad Rahmati and Lin Zhong. 2013. Studying smartphone usage: Lessons from a four-month field study. *IEEE Transactions on Mobile Computing* 12, 7, 1417–1427.

Lenin Ravindranath, Jitendra Padhye, Sharad Agarwal, Ratul Mahajan, Ian Obermiller, and Shahin Shayandeh. 2012. AppInsight: Mobile app performance monitoring in the wild. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation*. 107–120.

Paul R. Rosenbaum and Donald B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70, 1, 41–55.

Ardalan Amiri Sani, Zhiyong Tan, Peter Washington, Mira Chen, Sharad Agarwal, Lin Zhong, and Ming Zhang. 2013. The wireless data drain of users, apps, and platforms. *ACM SIGMOBILE Mobile Computing and Communications Review* 17, 4, 15–28.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments—debiasing learning and evaluation. In *Proceedings of the 33rd International Conference on Machine Learning, Vol. 48*. 1670–1679.

Kent Shi and Kamal Ali. 2012. GetJar mobile application recommendations with very sparse datasets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 204–212.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*. 623.

Yuan Tian, Meiyappan Nagappan, David Lo, and Ahmed E. Hassan. 2015. What are the characteristics of high-rated apps? A case study on free Android applications. In *Proceedings of the 31st IEEE International Conference on Software Maintenance and Evolution*. 301–310.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 58, 1, 267–288.

Chad Tossell, Philip T. Kortum, Ahmad Rahmati, Clayton Shepard, and Lin Zhong. 2012. Characterizing Web use on smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2769–2778.

Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering*. 14–24.

Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR Conference*. 115–124.

L. Wei, Y. Liu, and S. C. Cheung. 2016. Taming Android fragmentation: Characterizing and detecting compatibility issues for Android apps. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. 226–237.

Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. 2011. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement*. 329–344.

Yu Zheng and Xiaofang Zhou. 2011. *Computing with Spatial Trajectories*. Springer Science & Business Media.

Received June 2016; revised October 2016; accepted November 2016