PRADA: Prioritizing Android Devices for Apps by Mining Large-Scale Usage Data

Xuan Lu[1] Xuanzhe Liu[1‡] Huoran Li[1] Tao Xie[2] Qiaozhu Mei[3] Dan Hao[1] Gang Huang[1] Feng Feng[4]
[1]Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, PRC
[2]University of Illinois at Urbana-Champaign, [3]University of Michigan,[4]Wandoujia Lab, Beijing, China
{luxuan, xzl, lihuoran}@pku.edu.cn, taoxie@illinois.edu
qmei@umich.edu, {haodan, hg}@pku.edu.cn, jackfeng@wandoujia.com

ABSTRACT

Selecting and prioritizing major device models are critical for mobile app developers to select testbeds and optimize resources such as marketing and quality-assurance resources. The heavily fragmented distribution of Android devices makes it challenging to select a few major device models out of thousands of models available on the market. Currently app developers usually rely on some reported or estimated general market share of device models. However, these estimates can be quite inaccurate, and more problematically, can be irrelevant to the particular app under consideration. To address this issue, we propose **PRADA**, the first approach to **pr**ioritizing **A**ndroid **d**evice models for individual **a**pps, based on mining large-scale usage data. PRADA adapts the concept of operational profiling (popularly used in software reliability engineering) for mobile apps – the usage of an app on a specific device model reflects the importance of that device model for the app. PRADA includes a collaborative filtering technique to predict the usage of an app on different device models, even if the app is entirely new (without its actual usage in the market yet), based on the usage data of a large collection of apps. We empirically demonstrate the effectiveness of PRADA over two popular app categories, i.e., *Game* and *Media*, covering over 3.86 million users and 14,000 device models collected through a leading Android management app in China.

Categories and Subject Descriptors

D.2.9 [**Management**]: Software quality assurance (SQA)

Keywords

Mobile apps, Android fragmentation, prioritization, usage data

‡Xuan Lu and Xuanzhe Liu are both first authors, and contributed equally to the work.
§Corresponding author.

ICSE '16, May 14-22, 2016, Austin, TX, USA
 ISBN 978-1-4503-3900-1/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2884781.2884828

1. INTRODUCTION

The wide adoption of smartphones and tablet computers has triggered a surge of developing mobile applications, a.k.a, apps, in recent years. As of 2015, millions of apps have been developed and made available in app marketplaces such as the Apple Store and Google Play, which have received billions of downloads. Numerous app developers have profited from the revenues generated by the downloads and usages of their apps.

Compared to the iOS and Windows platforms, which have a rather fixed set of device models[1], the Android platform is adopted by a diverse set of device manufacturers and models. Indeed, the openness and flexibility of the Android platform have greatly contributed to its popularity: the Android platform holds more than 80% of the smartphone market share. Famous brands, such as Samsung, HTC, Motorola, and Lenovo, have developed numerous device models using Android. Meanwhile, most small and medium device manufacturers also adopt Android as their platforms.

The heavily fragmented distribution of Android device models is noticeable. It is reported by OpenSignal [6] that there have been more than 20,000 Android device models on the market up to the year 2014. Such a fragmentation brings significant challenges to software engineering practices for mobile apps, such as the design, development, maintenance, quality assurance, and revenue generation [20, 31]. A recent study [1] (in 2013) shows that 94% of app developers who avoid the Android platform cited fragmentation as the main reason. Developers have to take into account device-specific factors such as screen sizes, resolution levels, and other hardware specifications. An app (especially a game app) can run smoothly on high-end device models, which have powerful computation power and high resolution, but may run sluggishly or improperly on lower-end devices. Developers need to conduct extensive testing and other quality assurance activities to validate the functionality and usability (such as the GUI effects) of their apps on multiple device models. The fragmentation also influences the revenue, of which in-app advertisement (or *ads*) is an important channel, especially for some types of apps (e.g., game and media apps). To more accurately target the audience, device-specific ads are often preferred. For example, Facebook customizes mobile ads according to device model types [4] since 2014. Developers would like to know through which device models they can gain more users and more ads-clicking opportunities, so that they can invest their effort in customizing the

[1]A device model refers to the specific model of devices that share the same hardware specification.

ads on those models, e.g., designing banners of proper sizes or placing videos at proper positions on the screen. As it is unrealistic to customize for every Android device model, selecting the major device models to focus on is quite important to Android app development. A suboptimal selection may cause to waste money and human effort, miss potential bugs, or even lose revenue, etc.

Most developers prioritize a small number of device models (normally less than ten) based on their market shares. These developers rely on the widely accessible market share reports or predictions, such as those provided by AppBrain [2]. However, the market share of a device model is usually calculated based on how many devices are sold instead of how they are used, which is what the developers really care about. More importantly, the market share of a device model may not be relevant to particular apps. Indeed, it is not uncommon that an app is heavily installed on less popular device models and less preferred by those who use a majority model. Even if an app is installed on a device, it may be frequently or barely used [32]. To make the right decision, the developers need an accurate estimate of how their apps are actually used on different device models.

To address this issue, we propose a novel approach, named **PRADA** (**Pr**ioritizing **A**ndroid **D**evices for **A**pps), to prioritizing major device models of a given app. Rather than counting the number of devices installing an app, PRADA utilizes a different signal – how the app is actually used on the devices. The key intuition of PRADA is derived from the concept of *operational profile* [28], a concept popularly used in software reliability engineering, which is a quantitative representation of how a system is used. PRADA assumes that a device model is of a higher priority for an app if the app is consumed more intensively by the users using that device model. PRADA then builds a data mining model that accurately predicts the major device models for every app, even if it is newly launched on the market and the actual usage data is unavailable. Based on the predictions, PRADA recommends a ranked list of device models that should be prioritized for an app.

We evaluate the effectiveness of PRADA through a real world data set collected by a leading Android app marketplace in China, Wandoujia [9]. Wandoujia provides its native management app to facilitate searching, downloading, and updating apps. In addition, the management app also provides an interface to monitor the daily usage of the apps installed on a device. We choose the top 100 popular apps in two app categories, i.e., *Game* and *Media*. These apps (100 apps from each category) cover 3,861,444 users and 14,709 device models. We adopt *browsing time* as the typical metric of usage data, as it indicates the total time that users interact with a specific app under network. We then evaluate the significance of PRADA-selected device models accounting for the actual browsing time. Empirical results show that, compared to the baseline of marketshare, PRADA selects device models with top *browsing time*, i.e. device models that can maximize the coverage of *browsing time*, more accurately.

In summary, we make the following main contributions:

- We propose the first approach to prioritizing Android device models by mining app usage data collected from Android devices.
- We develop a collaborative filtering technique that predicts major device models for a *new* app based on the usage of other apps with similar functionalities.
- We present the largest study to date on prioritizing device models using a real-world data set.
- We conduct experiments to evaluate the effectiveness of PRADA and its related approaches.

The rest of this paper is organized as follows. Section 2 describes the related work of Android device prioritization. Section 3 presents the overview of PRADA including its key ideas and workflows. Section 4 illustrates PRADA through an example case and Section 5 presents an evaluation. Section 6 presents the findings and implications for developers. Section 7 discusses threats to validity of our study. Section 8 concludes the paper with future work.

2. RELATED WORK

We next present a summary of existing studies on Android device fragmentation, operational profiles, and analyses of app usage data.

2.1 Android Fragmentation

Compared to software development for PC, one unique challenge of developing Android apps is how to cope with the heavy fragmentation of Android devices. Halpern *et al.* [19] make a careful analysis of the market fragmentation caused by hardware specifications and OS versions, and propose a capture-and-replay approach for testing. Park *et al.* [31] propose two approaches to handle Android fragmentation at the code level and the API level. Han *et al.* [20] analyze the bug reports related to the two popular mobile-device vendors, `HTC` and `Motorola`, and propose an approach for tracking fragmentation using feature analysis on project repositories. Khalid *et al.* [22] leverage user reviews to focus on proper Android devices for app testing. They collect about 100 thousand user reviews of 99 free game apps on Google Play. However, Google Play has recently shut down the API of browsing user reviews for all device models, except for the user reviews from specific device models associated with a user's Google account; such constraint may introduce a considerable bias when leveraging user reviews. Another concern is the potential subjectiveness and biases in user reviews. For example, it is reported that users from different countries may behave very differently in writing reviews [25].

Some existing industrial cloud-based testing services, such as Testin [8] and AppthWack [3], provide remote services and offer a sufficient coverage of device models. However, the cost ranges from approximately one dollar per every 15-minute use of a device, leading to a very high expense if one wants to test many device models.

2.2 Operational Profiles

The concept of operational profile is widely used in software engineering, especially software reliability engineering and software testing [11, 16]. Musa [28] defined an operational profile as "***a quantitative representation of how a system will be used***". It models how users execute a system, specifically the occurrence probabilities of function calls and the distributions of parameter values. Such a description of the user behavior can be used to generate test cases and to direct testing to the most used functions. For example, with a software system, if operation A occurs in 60 percent of the time, B occurs in 30 percent, and C occurs in 10 percent, then the profile is [A, 0.6...B, 0.3...C, 0.1]. Descriptions of the user behavior as in an operational

profile can also be used for other purposes besides software testing [10]. The performance and correctness of the system can be analyzed, and the system can be effectively adapted to specific user groups. For example, Mobahser *et al.* [27] use operational profiles of Web-based systems for personalization.

An operational profile helps improve the communication between customers and developers, and make the developers think more deeply about the features of interest and their importance to the customers. In contrast, if developed early, an operational profile may be used to prioritize operations under development, so that more resources are invested on more important operations.

2.3 App Usage Data Analysis

From the perspective of software engineering, usage data of apps is a typical operational profile. Understanding how an app is used by real-world users is also important to improve the development of the app. One common approach is to conduct a field study with typically a small group of users, as proposed in *LiveLab* [39, 34, 33]. Many other studies have been conducted in a similar fashion, for reporting the diversity of cellular usage from different user groups, different app categories, etc. [21, 15, 12, 13, 17]. To collect usage data, some researchers develop third-party apps and deploy them as system-level services, such as AppInsight [35] and AppJoy [41]. The app usage data can include detailed information such as the total launch time, the traffic volume, and the session lengths to investigate user preferences and interests in depth. However, these apps are not widely adopted and their users are usually at the scale of hundreds. Compared to these field studies, our approach relies on a commercial app, Wandoujia, which has been widely deployed among over 250 million users. Although most Wandoujia users are from China, our recent work [24] conducts a series of studies of understanding user behaviors from multiple dimensions such as app popularity, network usage, and price sensitivity, and validates the general findings from previous efforts made over other popular app stores [32] and the traces at tier-1 cellular network in US [40]. The results show that conducting studies using Wandoujia data can reduce the threats caused by user selection bias. In addition, such a longitudinal data set collected from about 4 million anonymized users can make many more comprehensive analyses feasible.

3. THE PRADA APPROACH

In this section, we present the PRADA approach of prioritizing device models by using large-scale usage data. The key idea of PRADA is that, *a device model should be given higher priority for an app, if the device model accounts for more user activities.* Naturally, when an app is already used by many users on many devices, estimating the length of time it is used or the number of user interactions is an easy task. We are motivated, however, to explore whether we can make accurate estimates for a new app, which has not yet reached a critical mass of users or even has not yet released. Market targeting and prioritized testing at this stage are particularly important and challenging for the developers. PRADA relies on the usage data of apps on specific device models collected by Wandoujia.

3.1 Wandoujia

The app usage data are collected through a commercial Android app management tool developed by a leading Android marketplace in China, called the Wandoujia. Wandoujia was founded in 2009 and has grown into one of the largest Android app marketplaces in the world, with over 250 million users and 1.5 million free Android apps[2] as of the year 2015. Each user is associated with at least one Android device, either a smartphone or a tablet computer.

Figure 1: Screenshots of advanced settings in the Chinese version of the Wandoujia management app (the advanced settings are not supported in the current English version). (a) is the homepage of the Wandoujia management app, where users can navigate to "*settings*" by clicking on the text circled by red; (b) refers to the setting of background management services, highlighted by the red rectangle; (c) refers to the option of allowing Wandoujia to collect the data of network activities or not.

Wandoujia provides a native management app, through which users can manage the apps on their devices, e.g., downloading, searching, updating, and uninstalling apps. Users can also rate/review apps. Beyond these basic features, the Wandoujia management app is developed with some optional features that can monitor and optimize system-wide activities. These features include network activity statistics, permission monitoring, content recommendation, etc. All features are developed upon Android system APIs and do not require the "root" privilege. Users can opt in and out these features. For example, as shown in Figure 1, tracking the network statistics is an explicit option for end users, and therefore our analysis is made on only those users who agree to share and upload their usage data. However, it should be noted that these features are supported in only the **Chinese version** of Wandoujia.

We obtain three months of app usage data collected from July 1st, 2014 to September 30th, 2014. The data cover 4,775,293 unique users, 16,602 device models, and 238,231 apps. Every device has a unique `IMEI` identifier, and the corresponding information of the device model is also captured.

• **User Privacy Protection.** We take a series of steps to preserve the privacy of involved users in our data set. First, all raw data collected for this study are kept on the Wandoujia data-warehouse servers (which live behind a company firewall). Second, our data-collection logic and analy-

[2]Most apps released on Wandoujia can also be found on other app stores such as Google Play.

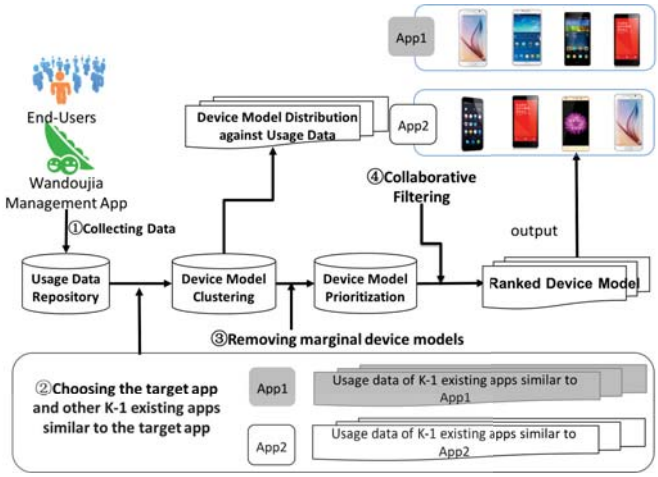


Figure 2: Overview of the PRADA Approach.

sis pipelines are governed by Wandoujia employees[3] to ensure compliance with the commitments of Wandoujia privacy stated in the *Term-of-Use* statements. Third and the most significantly, Wandoujia employees anonymize the user identifiers before any data analysis. Only aggregated statistics are produced for the users covered in our study period. Finally, we obtain the approval from the research ethnics committee of School of Electronics Engineering and Computer Science in Peking University to conduct this research.

3.2 PRADA: in a Nutshell

Based on the collected data, PRADA aims to recommend the device models that should be prioritized for **a new app**, which has not been used by many users and therefore its usage data is either unavailable or untrustable. The basic idea is to predict the "expected" usage of this new app based on the characteristics of the device models of **existing** apps that are *similar* to the new app. We assume that the usage data of this new app cannot be attained or is not informative to use. We describe the key components and workflows of PRADA, as illustrated in Figure 2.

• **Usage-Data Collection.** PRADA is general by leveraging various usage data from an operational profile. Currently, PRADA employs the Wandoujia management app, which provides the interface for collecting multiple types of usage data of an app on a device, such as the lengths of in-app network sessions, the volume of network traffic, the energy drain, and the user reviews. Developers can choose one or more types of usage statistics of their interest as features for prioritizing device models. For example, some developers may want to know which device models account for longer time of network usage and design device-specific ads to target that audience; some developers may distribute more testing efforts on device models that contribute to more negative user reviews, etc.

• **Similar-App Selection**. PRADA employs the idea of *collaborative filtering*: for an app whose major device models need to be predicted, PRADA relies on the usage data from a set of **existing similar** apps. In practice, PRADA is compatible to different types of app *similarity* measurement [14], e.g., the name, textual description, code, library, and category. In this paper, we adopt the *category* of Wandoujia's classification system[4]. For each new app, we choose some existing apps with usage data from the same category and perform the following steps. Although the category may be conceptually coarse-grained, using the category can help achieve sufficient effectiveness as shown in our evaluation.

• **Device-Model Clustering**. Given the selected type(s) of usage statistics, PRADA conducts an offline analysis of the device model distribution of each app from the selected similar apps. In this step, PRADA summarizes all devices that have ever produced the selected usage data of the app, and clusters them according to their device models (specified in their identifiers). For example, assuming that there are 1,000 *Samsung Galaxy S3* smartphones using a specific app, we aggregate together the selected usage data generated over them.

• **Device-Model Prioritization.** As the core component of PRADA, device-model prioritization produces a ranked list of device models based on predicting the selected type of usage statistics. To prioritize the device models, PRADA is designed based on the following two rationales.

- **Pareto Distribution.** It is well known that in many situations, a larger portion of effects comes from a smaller percentage of the causes (or roughly 80%-20% rule), known as the Pareto distribution [30]. We hypothesize that a similar distribution is still valid in the distribution of usage data contributed by device models. If such a hypothesis holds, we can remove a large portion of device models to significantly reduce the space of prioritization.
- **Collective Intelligence.** The second rationale is that similar apps may share similar distributions of device models. Hence, we can leverage the knowledge derived from a large number of existing similar apps to predict the to-be-prioritized device models for a new app. This rationale is shared by collaborative filtering [36], which is a major approach underlying most recommender systems [37]. Various collaborative filtering techniques adopt different notions of "similarity".

Based on the preceding two rationales, the goal of PRADA is to recommend a small set of device models that account for a desired coverage of usage data of a new app, given by the distribution of usage data of existing similar apps.

3.3 Effectiveness Metrics

Before presenting the details of PRADA, we introduce some metrics to evaluate the effectiveness of any concrete algorithm that PRADA incorporates.

The first metric is the *Device Model Hit*, which is the number of recommended device models (by PRADA for an app) that are "actually" among the top device models of the app (observed after the app is deployed and used by users).

• **Definition 1:** Given the number of device models (denoted as N) to be recommended and the number of the existing similar apps (denoted as K-1), the **Device Model Hit** is the size of the overlap between the recommended N device models $\{D_1^{'}, D_2^{'}, ..., D_N^{'}\}$ and the actual top N device models $\{D_1, D_2, ..., D_N\}$. Formally, we can define

$$Device\ Model\ Hit(N,K) = |D_{N_{recommended}} \bigcap D_{N_{actual}}| \quad (1)$$

[3]The last author is the co-founder and CTO of Wandoujia. He supervises the process of data collection and de-identification.

[4]Wandoujia's classification system is based on criteria including the developer's annotation, textual description, and some code-level analytics. The details of the classification system is out of the scope of this paper.

Suppose the actual top 5 device models are $\{D_1, D_2, D_3, D_4, D_5\}$. If PRADA recommends $\{D_2, D_3, D_1, D_5, D_6\}$, the *device model hit* is 4.

The *Device Model Hit* measures how many recommended device models are valid, but does not distinguish among these recommended items. In practice, the ranking of the recommendations is usually important so that the developers can prioritize on any number of device models based on their budget. We use the metric of *Average Precision* (abbreviated as AP in the rest of this paper) to evaluate the effectiveness of the ranking of device models; such metric has been widely used in evaluating search engines [26].

• **Definition 2: Average Precision (AP)** of the selected N device models $\{D_1^{'}, D_2^{'}, ..., D_N^{'}\}$ is the average precision of $D_i^{'}$ $(1 \leqslant i \leqslant N)$ against the actual top N device models selected for their top ranking in the contribution of the usage data:

$$AP = \sum_{i=1}^{N} Precision(D_i^{'})/N \tag{2}$$

$Precision(D_i^{'})$ denotes the precision at cut-off position i where the device model $D_i^{'}$ stays in the ranked device model list. $Precision(D_i^{'})$ is equal to 0 when $D_i^{'}$ is not found in the actual top N device model list.

We take a simple example to illustrate AP. Suppose that the actual top 5 device models (the ground truth) with the most usage data are $\{D_1, D_2, D_3, D_4, D_5\}$, where each device model is ordered according to its contribution to usage data. When the set of ground truth is fixed, the order of the device models is no longer concerned. Assume that the top 5 device models recommended by PRADA are $\{D_1, D_2, D_3, D_4, D_6\}$, then the AP is computed as $((1/1 + 2/2 + 3/3 + 4/4 + 0))/5 = 0.8$.

Note that two sets of device models that have the same *Device Model Hit* may vary much in terms of AP. Consider another list of device models $\{D_6, D_1, D_2, D_3, D_4\}$, whose *Device Model Hit* is still 4. However, its AP is $((0 + 1/2 + 2/3 + 3/4 + 4/5))/5 = 0.54$.

The third metric that we use is the *Usage Data Coverage*, which measures how much the recommended device models cover the entire set of usage by the actual top N devices of the app. Such a metric can reflect how much the recommended device models can contribute to the usage data of interest, and therefore has an indication of the potential opportunity of revenue.

• **Definition 3: Usage Data Coverage** is the percentage of aggregated usage statistics of recommended N device models over the aggregated usage statistics of actual top N device models. Formally, we can define

$$Usage\ Data\ Coverage(N, K) = \frac{Usage\ Data(D_{N_{recommended}})}{Usage\ Data(D_{N_{actual}})} \tag{3}$$

A successful instantiation of PRADA is expected to achieve high scores of the three metrics. Next we illustrate how to use PRADA to prioritize device models.

4. TIME-SHARE DRIVEN PRIORITIZATION

In this section, we illustrate PRADA through an example case focusing on the in-app browsing time collected from Wandoujia while using the *category* to determine similar apps.

4.1 Browsing Time on an App

As shown in Figure 1, the Wandoujia management app provides a system-wide service for recording daily network activity statistics of each app, for both Wi-Fi and cellular (2G/3G/LTE). The Wandoujia management app does not record the details of each interaction session. Instead, it records the total daily access time generated from both Wi-Fi and cellular network, by aggregating the time across TCP flows generated by an app. The Wandoujia management app treats the network access time generated from foreground and background, respectively.

Foreground access time is computed only when a user browses an app, i.e., the app is currently active on screen. We can roughly measure *how long a user is really "online" when she interacts with the app*, by aggregating the ***foreground*** access time of Wi-Fi and cellular. We call such aggregated time as "***browsing time***" (unless stated otherwise, we exchangeably use "time" and "browsing time" in the rest of this paper). Such a type of operational profile could be a useful indicator for app developers. For example, in-app advertisement is an important revenue for mobile app developers [18]. Furthermore, as suggested by Facebook [4], it becomes popular to customize device-specific ads [29]. A common rationale used by online advertisement is that the longer a user stays on the site, the more probably she will click through the ads. Although the browsing time cannot capture the offline usage of an app, such metric is valuable, as apps with online app usage are increasingly widespread and important.

Definition 4: Formally, we define the browsing time for an app A contributed by device model D as follows:

$$Time(D \rightarrow A) = \sum Time(d_i \rightarrow A) \tag{4}$$

Here, D denotes a specific device model and d_i is a particular device that belongs to device model D, i.e., $d_i \in D$.

To reflect the importance of device models that account for browsing time, we introduce the metric ***time share*** as the indicator of ranking. The *time share* of a device model is the percentage of *browsing time* consumed by the specific model to that consumed by all the models that use the app.

Definition 5: Formally, we define the time share as follows:

$$Time\ Share(D_j \rightarrow A) = \frac{Time(D_j \rightarrow A)}{\sum Time(D_k \rightarrow A)} \tag{5}$$

Here, $Time(D_j)$ is the time spent on a specific device model and $\sum Time(D_k)$ is the total time spent on all device models, i.e., all users of the app. The higher time share a device model D_j holds, the more time users use the app on D_j.

Suppose that we select N device models, we can use the metric of *Time Share Coverage* to concretize the *Usage Data Coverage* defined in equation 3 to measure the effectiveness of PRADA.

$$\begin{aligned} Time\ Share\ Coverage &= \frac{\frac{Time(D_{N_{recommended}})}{\sum Time(D_k \rightarrow A)}}{\frac{Time(D_{N_{autual}})}{\sum Time(D_k \rightarrow A)}} \\ &= \frac{Time\ Share(D_{N_{recommended}})}{Time\ Share(D_{N_{actual}})} \end{aligned} \tag{6}$$

4.2 Collaborative Filtering by Time Share

Recall that the PRADA approach accepts inputs as (1) the target app, and (2) N as the number of device models

that app developers would take into account. We predict the rank of time share from device models for this target app based on the "known" similar top K-1 apps in the same app category. Then the collaborative filtering technique outputs the top N major device models.

More specifically, we derive DM as the set of device models that at least one app of the K-1 apps uses. Then, across all apps in the K-1 apps, we compute the aggregated browsing time for each device model in DM. Finally, we sort the device models in DM by their aggregated browsing time, and recommend the top N device models as output for the target app.

To evaluate the effectiveness of the collaborative filtering technique, we then perform the process of Leave-One-Out Cross-Validation (**LOOCV**) [23]. LOOCV uses one observation as the validation set and the remaining observations as the training set, and then repeats as each observation has served as the validation set. Thus, for each of the K apps, we use the browsing time of the remaining K-1 apps to predict the top device models for the app. In particular, we use LOOCV, rather than hold out cross validation, since most apps have far less browsing time than popular apps.

For each app category, we select K apps and run the Algorithm 1: DH as a list of $\langle app, device\ model\ hit \rangle$, TC as a list of $\langle app, time\ share\ coverage \rangle$ and AP as a list of $\langle app, AP \rangle$. For the inputs, we take a list of K apps (AL), a total set (Ω) of $\langle device\ model, time \rangle$ for all the K apps. In each iteration of the algorithm, the app $\mathbb{A}$ is treated as a "new" app (validation set) and the remaining K-1 apps as the "existing" apps (training set). For each app $\mathbb{A}$, we first obtain the $\langle device\ model, time \rangle$ set ($\omega_{\mathbb{A}}$) from Ω to find the actual top N device models (TD). Then we use the other K-1 apps to recommend N apps (TD') for app $\mathbb{A}$. Using TD and TD' of $\mathbb{A}$, we can calculate the intersection results as *device model hit*, and calculate AP using equation 2. Finally, for each device model $\mathbb{D}$ in TD', we update the *time* with the *time* from $\omega_{\mathbb{A}}$ and calculate the *time share coverage* using equation 6.

To better illustrate the algorithm, we present an example of *Game* apps. Suppose that N is 10 and K is 100 in our example, respectively. We instantiate $\mathbb{A}$ as app `Modoomarble`[5], a popular game from `Tencent` [7]. By aggregating the *browsing time* of device models from the other popular 99 *Game* apps, we obtain top 10 device models as the recommendations for `Modoomarble`. Comparing the device models with the "actual" top 10 device models of `Modoomarble`, we find that the first 8 device models are overlapped, and thus the AP is 0.8. The "actual" top 10 device models account for 76.3% time share of the total time of `Modoomarble`, while our selected 10 device models account for 74.2%. More specifically, the two distinct device models selected by our technique, *Xiaomi 2s* and *Xiaomi 3*, do not occur in the "actual" top 10 device models of `Modoomarble`, and these two device models actually contribute 0.5% and 0.7% time share, respectively. In contrast, the two device models missed by our technique (*HTC One* and *Nexus 5*) account for 2.3% and 1.1% time share, respectively. However, the time share coverage of our technique (against the time share from "actual" top 10 device models) can reach 97.2% (i.e., 74.2%/76.3%).

[5]http://www.wandoujia.com/apps/com.tencent.modoomarble

Algorithm 1: Device model hit (DH), time share coverage (TC) and average precision (AP) against top N device models with K apps in the same category

Input: AL, Ω, N, K
Output: DH, TC, AP
for *each app* $\mathbb{A}$ *in* AL **do**
  $\omega_{\mathbb{A}} = \Omega(\mathbb{A})$;
  //get $\langle device\ model, time \rangle$ for all device models
  //using app $\mathbb{A}$
  $TD_{\mathbb{A}} = max_N(\omega_{\mathbb{A}})$;
  //get top N device models most used by $\mathbb{A}$
  $AL' = AL - \{\mathbb{A}\}$;
  //AL' is the list of apps except $\mathbb{A}$
  $\omega' = \bigcup_{\mathbb{A}' \in AL'} \Omega(\mathbb{A}')$;
  //get $\langle device\ model, time \rangle$ for all device models
  //using apps in AL'
  //*time* of same *device model* is added up
  $TD' = max_N(\omega')$;
  //get top N device models most used by AL'
  //as the prediction for $\mathbb{A}$
  $DH(\mathbb{A}) = |TD_{\mathbb{A}} \bigcap TD'|$;
  //the intersection of predicted N device models and
  //the actual ones
  $AP(\mathbb{A}) = \frac{\sum_{i=1}^{N} |TD(\mathbb{A})_{top\ i} \bigcap TD'|/i}{N}$;
  //get the AP of the prediction $\mathbb{A}$
  //$TD(\mathbb{A})_{top\ i}$ means the subset constituted of the
  //top i items in set $TD(\mathbb{A})$
  for *each device model* $\mathbb{D}$ *in* TD' **do**
    $TD'(\mathbb{D}) = \omega_{\mathbb{A}}(\mathbb{D})$;
    //get the time that $\mathbb{D}$ (in TD') spent on $\mathbb{A}$
  end
  $TC(\mathbb{A}) = \sum_{\mathbb{D}} TD'(\mathbb{D}) / \sum_{\mathbb{D}} TD_{\mathbb{A}}(\mathbb{D})$;
  //get the *time share coverage* of the recommended
  //device models for $\mathbb{A}$
end

5. EVALUATION

In this section, we evaluate the PRADA approach over two typical types of apps that care about browsing time, *Game* and *Media*. More specifically, we intend to answer two research questions:

- **RQ1**: *How many device models account for the majority of the browsing time?* To demonstrate the *Pareto distribution* underlying PRADA, we perform a characteristic analysis of the time share distribution of all device models for apps from two popular categories.
- **RQ2**: *How effectively can PRADA identify major device models for a new app given that developers have no knowledge about this app's actual usage?* To demonstrate how the *collective intelligence* can help device model prioritization, we explore the utility of the time share to help app developers select N major device models of a new app whose time share usage is entirely unknown. Here, we apply our collaborative filtering technique to predict the time share of the app based on the browsing time of top K-1 apps from the same app category.

5.1 Device Model Distribution

We first address **RQ1**, i.e., *how many device models account for the majority of the browsing time?* This question

corresponds to the *Pareto Principle* of the time share distribution of all device models for an app and can indicate the number of major device models for an app.

In our three-month dataset, we have a total of 16,602 distinct Android device models. It would seem to be quite challenging and time-consuming for app developers to work across the whole set of existing Android device models. Hence, PRADA first summarizes the distribution of device models of an app, according to the browsing time.

In our current approach, we take the apps from the same category as "similar" apps. There are 14 categories defined by Wandoujia, such as *Communication*, *Tools*, *Media*, and *Game*. We aggregate the browsing time of all apps belonging to the same category. This clustering process can better organize a large number of apps and identify which categories contribute more browsing time. Results show that apps from some categories take up a lot of browsing time. We refer to such apps as *networked apps*.

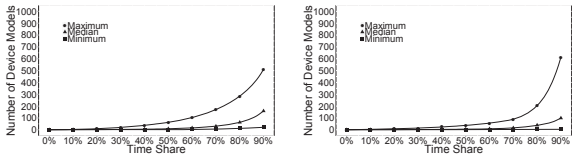
We then investigate the time share distribution at the level of every individual app. We choose two networked app categories *Game* and *Media*.

Table 1: Number of device models and users that use top 100 apps from each of the two categories.

Category	# of Device Models	# of Unique Users
Game	11,538	2,159,238
Media	13,894	3,547,219

Table 1 shows the number of device models and unique users that use top 100 apps from each of the two categories. Each category has millions of recorded users. Such a scale of data can promise a comprehensive analysis. For each category, we choose the top 100 apps by their aggregated browsing time from all device models using the app. Figure 3 shows the maximum, median, and minimum number of device models that can account for X% time share of apps of each category, where X varies from 0 to 90. Although the maximum and minimum varies a lot for a specific app, the median number of device models that account for 80% time share is generally around 100.

Findings. The preceding result of **RQ1** validates the Pareto Principle of the distribution of device models contributing to browsing time. Although the distribution does not strictly follow the 80-20 rules in the Pareto principle, it still shows that a quite small percentage of device models can account for a quite large portion of browsing time. It indicates that developers of apps in these app categories can significantly narrow their selection space. Meanwhile, developers can approximately estimate how many device models they should use to reach a desired time share.



(a) Game (b) Media

Figure 3: The maximum, median and minimum numbers of device models covering X% time share for each app in the two categories.

5.2 Predicting Top Device Models

Based on the distribution of device models derived from **RQ1**, we can help reduce a large number of device models from the fragmented Android markets. However, such a finding is based on the assumption that the usage data is already known. For a new app that is to be put on shelf and is not associated with informative knowledge of usage data, how can we leverage the findings of **RQ1** to predict the device models to prioritize? Thus, we next address **RQ2**, i.e., *how effectively can PRADA identify major device models given that developers have no usage knowledge about this app?*

Such an evaluation is done by applying the collaborative filtering algorithm 1. We still investigate the 200 apps of the two categories, *Game* and *Media*. For an app $\mathbb{A}$ in one category, we assume that its usage data is unknown. We then apply the collaborative filtering algorithm to find the set of N device models with the most time share by leveraging the browsing time from the remaining K-1 apps in the same category. Here, we assign N as 10 while K as 100. We then report the distribution of *Device Model Hit*, *Time Share Coverage*, and *Average Precision* against the ground truth of $\mathbb{A}$. We repeat the process for all 100 apps for each category. In particular, to evaluate the overall precision of all apps, we consider the widely used metric **Mean Average Precision** (MAP), which is used to rank the results for a large number of queries [5]. Hence, we also compute MAP for apps from each category as follows:

$$MAP(N) = \sum_{j=1}^{K} AP_j/K \tag{7}$$

5.2.1 Results

We report the results of *Device Model Hit*, *Time Share Coverage*, and AP of top 10 device models that are predicted by PRADA for 100 apps in each category, i.e., N =10 and K=100.

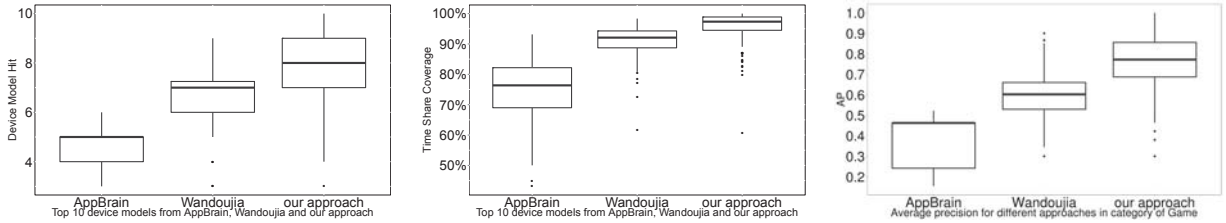
It is observed that PRADA works quite well for both categories. For apps in the *Game* category, the *Device Model Hit* is with 8 as the median, 10 as the maximum, and 3 as the minimum (shown in Figure 4(a)). Correspondingly, the boxplot of Figure 4(b) illustrates the *Time Share Coverage* from the 10 selected device models against the 10 "actual" top device models, with 97.4% as the median, up to 98.9% as the third quartile, and not less than 94.5% as the first quartile. The median of AP can reach 0.82 as median, and the MAP for *Game* apps is 0.75, which is a satisfactory score.

Similar to the *Game* category, the *Device Model Hit* results for *Media* are shown in Figure 5(a), with 8 as the median. In terms of *Time Share Coverage*, as shown in Figure 5(b), our technique can reach 96.7% as the median, 99.2% as the third quartile, and 90.2% as the first quartile, respectively. The median of AP is 0.79, and the MAP for *Media* apps is 0.72.

Although the core of PRADA is to leverage usage data, it is quite common that developers usually choose the device models from the most market share. However, such a selection is too coarse-grained, as it relies on only the number of active device models that have been on the market. More seriously, such selection may be inaccurate, with respect to a specific metric of the developers' interest. We make a simple comparison study between the top 10 device models that are predicted by PRADA and that are directly

Table 2: Top 10 device models with the most time share for two apps (Temple Run 2 and Xunlei Movie), and the selected device models by AppBrain, Wandoujia, and PRADA.

Top 10 device models in market		Top 10 device models for `Temple Run 2`		Top 10 device models for `Xunlei Movie`	
AppBrain	Wandoujia	Ground Truth	PRADA	Ground Truth	PRADA
Galaxy S3	Galaxy Note 3	Galaxy Note 3	Galaxy Note 3	Galaxy Note 3	Galaxy Note 3
Galaxy S4	Galaxy S4	Galaxy Note 2	Galaxy Note 2	Galaxy Note 2	Galaxy Note 2
Galaxy Note 3	Galaxy Note 2	Galaxy S4	Galaxy S4	Galaxy S4	Galaxy S4
Galaxy S5	Galaxy S3	MX 3	MX 3	MX 3	MX 3
Motorola Moto G	Galaxy Win	Galaxy S5	Galaxy S5	MX 2	Galaxy Mega 5.8
Galaxy S3 mini	Xiaomi 2s	Xiaomi 3	Galaxy Mega 5.8	Galaxy S5	Galaxy S5
Galaxy Tab 3.7	Xiaomi 3	Xiaomi 2s	MX 2	Galaxy Mega 5.8	MX 2
Galaxy Note 2	MX 3	Galaxy Mega 5.8	Galaxy S3	HTC One	Galaxy S3
Galaxy S Duos 2	Galaxy Mega 5.8	MX 2	Xiaomi 2s	Galaxy S3	Xiaomi 2s
Galaxy S2	Galaxy S2	Galaxy S3	Xiaomi 3	LG Nexus 5	Galaxy S2



(a) Device model hit (b) Time share coverage (c) Average precision

Figure 4: Comparison of *Device Model Hit*, *Time Share Coverage*, and *AP* by using market share and PRADA to recommend top 10 device models for *Game* apps.

obtained from the market share, respectively. The goal of such a comparison is two folds. First, we aim to demonstrate that simply relying on the market share is too coarse-grained and even inaccurate for prioritization, with respect to a specific metric of interest. Second, we aim to demonstrate that PRADA can achieve a satisfactory accuracy to help select device models.

To align with the time of our collected dataset, we choose the market share reports of Android device models of the 3rd quarter 2014, from the well-known AppBrain website [2]. AppBrain provides a *global market share* of device models. Due to the lack of detailed local market share of device models, we derive a *local market share* by aggregating the active users of a specific device model from Wandoujia.

We also compute the three metrics, *Device Model Hit*, *Time Share Coverage*, and *AP*, by using the device models with the most market share of AppBrain (in short as using AppBrain) and the ones with the most market share of Wandoujia (in short as using Wandoujia), respectively. From Figure 4(a), we can observe that for the *Game* apps, the 10 device models selected by using AppBrain reach 5 as the median of the *Device Model Hit*. The situation of using Wandoujia is a bit better than using AppBrain, reaching 7 as the median of the *Device Model Hit*. In other words, *Device Model Hit* by market share is worse than using PRADA, considering time share. In Figure 4(b), the median of time share covered by the device models by PRADA is 97.4%. In contrast, the value by using market share of Wandoujia is 92.1%, while the median of using AppBrain is 76.4%, which is the lowest. Additionally, the *MAP* value is 0.39 by using AppBrain, 0.62 by using Wandoujia, which is far away from 0.75 by using PRADA. Similar observations can be made for *Media* apps.

We show an example by two typical apps, which are also popular on Google Play, the `Temple Run 2`[6] and `Xunlei Movie`[7]. We list the the device models (1) with top 10 market share from AppBrain and Wandoujia (descending order of market share in Columns 1 and 2), respectively; (2) with the most actual browsing time for the given apps (descending order of time share in Columns 3 and 5); (3) ranked by PRADA (Columns 4 and 6), in Table 2.

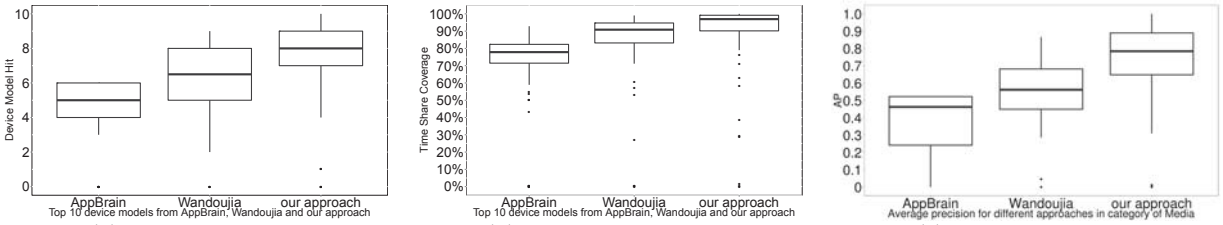
The top 10 device models with the most market share by AppBrain can cover only 5 out of the top 10 device models for `Temple Run 2` and `Xunlei Movie`, compared to the ground truth of browsing time. More specifically, AppBrain misses 5 of the top 10 device models for `Temple Run 2`. The device models missed by AppBrain are `MX 3, Xiaomi 3, Xiaomi 2s, Galaxy Mega 5.8`, and `MX 2`. In contrast, PRADA can hit all of the top 10 device models. For the `Xunlei Movie`, AppBrain misses 5 device models, i.e., `MX 3, MX 2, Galaxy Mega 5.8, HTC One`, and `LG Nexus 5`, while PRADA misses only `HTC One` and `LG Nexus 5`. The device models with the most market share from Wandoujia can have higher device model hit (8 for `Temple Run 2` and 6 for `Xunlei Movie`), but the hit ratio is still lower than PRADA.

As for *time share coverage*, it is 72.7% for `Temple Run 2` and 70.2% for `Xunlei Movie` by using the top 10 device models from AppBrain. In contrast, PRADA can achieve 100% and 97.5%. Using Wandoujia market share performs better than AppBrain but still worse than PRADA, which achieves 95.1% and 86.1% for the two apps, respectively.

The *AP* values by device models of AppBrain of the two apps are both less than 0.5. The *AP* is unsatisfactory (0.75 for `Temple Run 2` and 0.53 for `Xunlei Movie`) by the top de-

[6]https://play.google.com/store/apps/details?id=com.imangi.templerun2

[7]https://play.google.com/store/apps/details?id=com.xunlei.cloud



(a) Device model hit (b) Time share coverage (c) Average precision

Figure 5: Comparison of *Device Model Hit*, *Time Share Coverage*, and *AP* by using market share and PRADA to recommend top 10 device models for *Media* apps.

vice models from Wandoujia's market share. PRADA can reach the *AP* of 1.0 and 0.8, respectively. Such result illustrates that relying on only the market share is not accurate to predict the top device models against the actual operational profile of an app.

5.2.2 Findings

When addressing **RQ2**, we have some findings. When prioritizing device models, using the market share is not always sufficient or even accurate, with respect to a specific usage data, e.g., browsing time. In contrast, PRADA can more accurately identify device models on which users spend most browsing time.

Another finding is that the results derived from the Wandoujia market share are usually better than those from AppBrain. Such result indicates that the localization plays an important role of device model prioritization. Even relying on the market share, app developers targeting different areas would need to treat device models differently.

In both categories, there are some apps not well supported by our technique. For example, in *Media* apps, we find 3 apps whose *Time share Coverage* is 0 or close to 0, e.g., `HTC Album` (0%), `UMI Media Player` (0.6%), and `Android Music Player` (1.6%). Indeed, all top devices of `HTC Album` are developed by HTC. Most of the top device models of `UMI Media Player` and `Android Music Player` are local manufacturers from China. These two apps are also customized for these manufacturers and preloaded on their devices.

In summary, PRADA can accurately predict top device models, even if no informative usage data is given.

5.3 Discussion

The key idea of PRADA is to predict top device models of an app based on the usage of similar apps, and the measurement of *similarity* is quite general in PRADA. Currently, the selected "similar" apps are based on the *app category*. Indeed, generally assuming that apps from the same category to be similar is a bit simplistic and coarse-grained, but the classification system of Wandoujia has some effective criteria to categorize the apps. Our evaluation has already shown effective results achieved by using the app category. Indeed, some existing complex metrics proposed in the literature [14, 38] can be integrated into PRADA.

Although we choose only two popular app categories as illustrating examples, evidence actually supports that the top device models can vary a lot among different app categories. There are 14 categories defined by Wandoujia, i.e., *Business*, *Communication*, *Finance*, *Game*, *Lifestyle*, *Media*, *Mother_and_Baby*, *News*, *Productivity*, *Reading_and_Study*, *Shopping*, *Social*, *Tools*, and *Travel*. We perform the pairwise comparison of top 100 device models between categories, and the overlap is quite low. For example, the device model *Xiaomi 2s* is ranked as in the top 10 list of *Game*, but does not appear in the top 10 lists of 11 other app categories. *Xiaomi 3*, which is ranked as in the top 10 list of *Game*, is not ranked as in the top 10 lists of 13 other app categories. Thus, it is important to conduct category-specific recommendation of device models, as done in PRADA.

6. IMPLICATIONS

Based on a large-scale dataset collected from real-world users on interacting with Android apps, we have evaluated that operational profiles such as browsing time can accurately prioritize device models in the app categories of *Game* and *Media*. Although the specific results (e.g., which specific device models were top for an app or app category) from this study are useful for developers, device models and their usage are constantly evolved, and more recent usage data shall be used to re-apply PRADA in practice. Hence, we should focus on our general findings and methodologies beyond the specific results.

Relying on the market share is not always accurate, with respect to a specific metric of interest. In other words, more users do not always lead to more usage. Therefore, developers need to carefully explore and avoid uninformed investment on popular device models, which may not be significant with respect to the metrics of their interest.

The idea of leveraging the collective knowledge from large-scale usage data is feasible. By using the time share from major device models of other apps in the same app category, developers of new apps can accurately select major device models, even when the developers do not know which device models would heavily use their apps in the future. In addition, such idea can be extended to other types of usage data of an app (such as reported bugs, user reviews, traffic, and energy drain) over a specific device model.

Indeed, applying an approach such as PRADA requires a sufficient usage dataset that is *legally* collected from users. We plan to release a sample anonymized dataset to other researchers for further study. In addition, we plan to make PRADA available as an analytic service for assisting Android developers who publish or plan to publish their apps on Wandoujia.

7. THREATS TO VALIDITY

We have evaluated our approach in two main networked app categories. This section discusses the threats to validity in our evaluation.

A threat to validity includes the localization: using only Wandoujia (primarily in the China market). All users and

usage data are from Wandoujia, because other marketplaces such as Google Play do not release their usage data externally. We cannot validate the time-share-based technique over users and apps that are not included in the Wandoujia dataset. Although the large scale of dataset could provide comprehensive results to developers in China, developers from other regions cannot directly use the results as reference to predict device models for their apps used in their regions. However, the general idea of PRADA could be applicable, if developers have other published usage data such as AppJoy [41] and LiveLab [39, 34, 33]. However, the data should be at scale to enable comprehensive analysis. In future work, we plan to alleviate this threat by applying PRADA on other usage data beyond the Wandoujia dataset.

We apply PRADA based on a specific usage data, i.e., the browsing time that comes from foreground network access time in an app, indicating how long users interact with the app. The metric of time share can measure the importance of a device model for a specific app. Although most of these apps need the network connection, some of them do not always produce foreground network activities. Instead, their network activities are often performed in the background, such as downloading or updating. Some apps are mainly used offline, such as PDF readers. Therefore, our approach could not be generalized to all kinds of apps, if only the browsing time is used. However, the significance of using browsing time still remains because apps with online app usage are increasingly widespread and important. In addition, as long as using online app usage allows to preserve the ranking of device models, it can still achieve effective results.

The time sensitivity also impacts the effectiveness of PRADA. We limit only 3 months of data to evaluate our approach, i.e., from July to September. One reason for such study setup is that we want to compare the effectiveness of the time share and market share, where the latter is usually made quarterly. However, it is well known that the upgrade of smartphones is quite frequent, and users may buy new devices. Hence, the number of users for a device model may keep changing, correspondingly leading to the change of time share for an app. Therefore, for app developers, using our 3 months of dataset could not well predict the *currently* major device models. To alleviate this threat, possible solutions include performing our approach online by using the latest time share collected from the Wandoujia management app, or exploiting Wandoujia data covering a longer time span, e.g., 1 year or longer, to learn how user behaviors impact time share and improve the effectiveness of our approach. To this end, we need the server-side support of Wandoujia such as exposing online data-retrieval APIs, and avoid potential side effects and interferences to other online services of Wandoujia.

The release date of a device model can also impact the effectiveness of PRADA. PRADA relies on the real usage collected from substantial users within a reasonable period. If a device model is recently released and no enough usage data can be collected from this device model, the application scope of PRADA may be limited. Such a limitation cannot be completely overcome, as the device model is entirely new. However, the situation can be alleviated. One promising solution is to reduce the latency between the release date of the device model and the collection of usage data. To this end, PRADA can be deployed over the Wandoujia server, being timely sensitive to the usage data collected from this new device model.

To predict device models, we use only the *category* of apps defined by Wandoujia, to obtain similar K-1 apps for the collaborative filtering in PRADA. Therefore, the performance of PRADA currently relies on the accuracy of Wandoujia's category taxonomy of apps. As found in our previous work [24], a lot of apps from different categories may also have very high similarity. Besides the category information, more profiles of apps (such as the vendor information, user reviews, app requirements, and libraries) can be further leveraged to collect those apps sharing similar features with the given app under consideration. Some existing metrics [14, 38] are under consideration to be plugged into PRADA.

8. CONCLUSION AND FUTURE WORK

To address the challenges caused by Android device fragmentation, in this paper, we have presented the novel PRADA approach by using real-world usage data collected from a large number of users. PRADA includes a collaborative filtering technique to accurately predict major device models for a new app, given the usage data from existing apps with similar functionalities. We have evaluated PRADA by using the in-app browsing time, which indicates how much users interact with an app on a specific device model. In our study, we used 200 apps from two app categories (Game and Media), spanning three months and covering 3.86 million users and 14.71 thousand device models. Implications derived from our findings provide useful guidelines for Android app developers.

Since most of the data collected from Wandoujia is from China, a great deal of localization issues may account for differences compared to the global market. We plan to investigate the impact of localization on device model prioritization in future work. We also plan to further explore how to cluster device models at different granularities. Another ongoing effort is to measure the fragmentation impact caused by the great diversity of Android OS versions when applying PRADA.

Acknowledgment

This work was supported by the High-Tech Research and Development Program of China under Grant No.2015AA01A203, the Natural Science Foundation of China (Grant No. 61370020, 61421091, 61222203, 61572051, 61528201). Tao Xie's work was supported in part by National Science Foundation under grants no. CCF-1349666, CCF-1409423, CNS-1434582, CCF-1434596, and CNS-1513939, and a Google Faculty Research Award. Qiaozhu Mei's work was supported in part by the National Science Foundation under grant No. IIS-1054199. The authors would like to appreciate Le Lian for valuable suggestions on the application context of this work.

9. REFERENCES

[1] Android fragementation problem. http://www.greyheller.com/Blog/androids-fragmentation-problem.
[2] AppBrain. http://http://www.appbrain.com/.
[3] AppThwack. https://appthwack.com.
[4] Facebook: How do I run ads only on specific types of phones? https://www.facebook.com/business/help/607254282620194.
[5] Mean average precision. https://www.kaggle.com/wiki/MeanAveragePrecision.
[6] OpenSignal. http://opensignal.com/reports/2014/android-fragmentation/.

[7] Tencent. http://www.tencent.com.

[8] Testin. http://www.testin.cn.

[9] Wandoujia. http://www.wandoujia.com.

[10] S. Balsamo, A. D. Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, 2004.

[11] P. G. Bishop. Rescaling reliability bounds for a new operational profile. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 02*, pages 180–190, 2002.

[12] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage. In *Proceedings of the International Conference on Human-computer Interaction with Mobile Devices and Services, MobileHCI 11*, pages 47–56, 2011.

[13] M. Böhmer and A. Krüger. A study on icon arrangement by smartphone users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 13*, pages 2137–2146, 2013.

[14] N. Chen, S. C. H. Hoi, S. Li, and X. Xiao. SimApp: A framework for detecting similar mobile applications by online kernel learning. In *Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM 15*, pages 305–314, 2015.

[15] G. Chittaranjan, J. Blom, and D. Gatica-Perez. Mining large-scale smartphone data for personality studies. *Personal and Ubiquitous Computing*, 17(3):433–450, 2013.

[16] B. Cukic and F. B. Bastani. On reducing the sensitivity of software reliability to variations in the operational profile. In *Proceedings of the International Symposium on Software Reliability Engineering, ISSRE 96*, pages 45–54, 1996.

[17] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement, IMC 10*, pages 281–287, 2010.

[18] J. Gui, S. Mcilroy, M. Nagappan, and W. G. J. Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *Proceedings of the IEEE/ACM International Conference on Software Engineering, ICSE 15*, pages 100–110, 2015.

[19] M. Halpern, Y. Zhu, R. Peri, and V. J. Reddi. Mosaic: cross-platform user-interaction record and replay for the fragmented Android ecosystem. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 15*, pages 215–224, 2015.

[20] D. Han, C. Zhang, X. Fan, A. Hindle, K. Wong, and E. Stroulia. Understanding Android fragmentation with topic analysis of vendor-specific bugs. In *Proceedings of the Working Conference on Reverse Engineering, WCRE 12*, pages 83–92, 2012.

[21] J. Jung, S. Han, and D. Wetherall. Short paper: enhancing mobile application permissions with runtime feedback and constraints. In *Proceedings of the ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM 12*, pages 45–50, 2012.

[22] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan. Prioritizing the devices to test your app on: A case study of Android game apps. In *Proceedings of the SIGSOFT International Symposium on the Foundations of Software Engineering, FSE 14*, pages 610–620, 2014.

[23] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai*, 14(2):1137–1145, 1995.

[24] H. Li, X. Lu, X. Liu, T. Xie, K. Bian, F. X. Lin, Q. Mei, and F. Feng. Characterizing smartphone usage patterns from millions of Android users. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement, IMC 15*, pages 459–472, 2015.

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

[26] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[27] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data mining and knowledge discovery*, 6(1):61–82, 2002.

[28] J. D. Musa. Operational profiles in software-reliability engineering. *IEEE Software*, 10(2):14–32, 1993.

[29] S. Nath. Madscope: Characterizing mobile in-app targeted ads. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services, MobiSys 15*, pages 59–73, 2015.

[30] M. E. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[31] J. H. Park, Y. B. Park, and H. K. Ham. Fragmentation problem in Android. In *Proceedings of the International Conference on Information Science and Applications*, pages 1–2, 2013.

[32] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis. Rise of the planet of the apps: a systematic study of the mobile app ecosystem. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement, IMC 13*, pages 277–290, 2013.

[33] A. Rahmati, C. Tossell, C. Shepard, P. Kortum, and L. Zhong. Exploring iPhone usage: the influence of socioeconomic differences on smartphone adoption, usage and usability. In *Proceedings of the International Conference on Human-computer Interaction with Mobile Devices and Services, MobileHCI 12*, pages 11–20, 2012.

[34] A. Rahmati and L. Zhong. Studying smartphone usage: Lessons from a four-month field study. *IEEE Transactions on Mobile Computing*, 12(7):1417–1427, 2013.

[35] L. Ravindranath, J. Padhye, S. Agarwal, R. Mahajan, I. Obermiller, and S. Shayandeh. AppInsight: Mobile app performance monitoring in the wild. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation, OSDI 12*, pages 107–120, 2012.

[36] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the ACM conference on Computer supported cooperative work, CSCW 94*, pages 175–186, 1994.

[37] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[38] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan. What are the characteristics of high-rated apps? A case study on free Android applications. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution, ICSME 15*, pages 301–310, 2015.

[39] C. Tossell, P. Kortum, A. Rahmati, C. Shepard, and L. Zhong. Characterizing web use on smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 12*, pages 2769–2778, 2012.

[40] Q. Xu, J. Erman, A. Gerber, Z. M. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement, IMC 11*, pages 329–344, 2011.

[41] B. Yan and G. Chen. AppJoy: personalized mobile application discovery. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services, MobiSys 11*, pages 113–126, 2011.