# Do Users Care about Ad's Performance Costs? Exploring the Effects of the Performance Costs of In-App Ads on User Experience

Cuiyun Gao[a,b], Jichuan Zeng[b], Federica Sarro[c], David Lo[d], Irwin King[b] and Michael R. Lyu[b]

[a]*School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China*
[b]*Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China*
[c]*Department of Computer Science, University College London, United Kingdom*
[d]*School of Information Systems, Singapore Management University, Singapore*

ARTICLE INFO

*Keywords*:
In-app ads
User reviews
Ad costs
Empirical study

ABSTRACT

Context: In-app advertising is the primary source of revenue for many mobile apps. The cost of advertising (ad cost) is non-negligible for app developers to ensure a good user experience and continuous profits. Previous studies mainly focus on addressing the hidden performance costs generated by ads, including consumption of memory, CPU, data traffic, and battery. However, there is no research on analyzing users' perceptions of ads' performance costs to our knowledge.
*Objective:* To fill this gap and better understand the effects of performance costs of in-app ads on user experience, we conduct a study on analyzing user concerns about ads' performance costs.
*Method:* First, we propose RankMiner, an approach to quantify user concerns about specific app issues, including performance costs. Then, based on the usage traces of 20 subject apps, we measure the performance costs of ads. Finally, we conduct correlation analysis on the performance costs and quantified user concerns to explore whether users complain more for higher performance costs.
*Results:* Our findings include the following: (1) RankMiner can quantify users' concerns better than baselines by an improvement of 214% and 2.5% in terms of Pearson correlation coefficient (a metric for computing correlations between two variables) and NDCG score (a metric for computing accuracy in prioritizing issues), respectively. (2) The performance costs of the with-ads versions are statistically significantly larger than those of no-ads versions with negligible effect size; (3) Users are more concerned about the battery costs of ads, and tend to be insensitive to ads' data traffic costs.
*Conclusion:* Our study is complementary to previous work on in-app ads, and can encourage developers to pay more attention to alleviating the most user-concerned performance costs, such as battery cost.

## 1. Introduction

In-app advertising is a type of advertisement (ad) within mobile applications (apps). Many organizations have successfully monetized their apps with ads and reaped huge profits. For example, the mobile ad revenue accounted for 76% of Facebook's total sales in the first quarter of 2016 [21], and increased 49% year on year to about $10.14 billion in 2017 [20]. Triggered by such tangible profits, mobile advertising has experienced tremendous growth recently [1]. Many free apps, which occupy more than 68% of the over two million apps in Google Play [6], adopt in-app advertising for monetization. However, the adoption of ads has strong implications for both users and app developers. For example, almost 50% of users uninstall apps just because of "intrusive" mobile ads [2], which may result in a reduction in user volume of the apps. Smaller audiences generate fewer impressions (*i.e.*, ad displaying) and clicks, thereby making ad profits harder for developers to earn. Thus, understanding the effects of in-app ads on user experience is helpful for app developers.

User reviews serve as an essential channel between users and developers, delivering users' instant feelings (including the unfavorable app functionalities or annoying bugs) based on their experience. User review mining has been proven useful and significant in various aspects of app development, such as supporting app design [29], categorizing app issues for facilitating app maintenance [57, 41], and assisting app testing [28], etc. In this work, we resort to user reviews to identify users' perception about in-app ads.

Previous research has been devoted to investigating the hidden costs of ads, *e.g.*, energy [48], traffic [51], system design [27], maintenance efforts [31], and privacy [70]. For example, Gui *et al.* [31] found that the with-ads apps can lead to 30% more energy consumption than the corresponding no-ads versions. Relieving all the types of hidden costs is quite labor-intensive for app developers. Understanding users' concerns about these costs can help developers focus on the user-concerned cost types and reduce labor cost. Although there are studies using surveys to understand users' perceptions of mobile advertising, *e.g.*, interactivity [87], perceived usefulness [71], and credibility [14], there is still a lack of study on analyzing users' concerns about the practical performance costs of in-app ads. There are several challenges to perform this kind of analysis. First, collecting a large amount of user feedback that reflects ads' performance costs is intractable. According to Gui *et al.*'s manual analysis of 400 sample ad-reviews [32], only four (1%) of the reviews are related to mobile speed and one (0.25%) relates to battery. Moreover, only around 1% of collected reviews clearly

* J. Zeng is the corresponding author

✉ cygao@cse.cuhk.edu.hk (C. Gao); jczeng@cse.cuhk.edu.hk (J. Zeng); f.sarro@ucl.ac.uk (F. Sarro); davidlo@smu.edu.sg (D. Lo); king@cse.cuhk.edu.hk (I. King); lyu@cse.cuhk.edu.hk (M.R. Lyu)
ORCID(s):

deal with in-app ads. Second, users' concerns about ad costs are difficult to be quantified, where user behaviors (such as rating apps) should be well involved. Lastly, measuring the performance costs solely incurred by ads is difficult practically due to diverse usage patterns (*e.g.*, different ad viewing duration) from users.

In this paper, we try to overcome these challenges, and propose an approach, named RankMiner, to quantitatively measure user concern levels about specific app issues. Note that RankMiner can measure users' concerns about any specified app issues besides the performance-related ones studied in this paper. To verify the effectiveness of RankMiner, we choose CrossMiner [42], an issue ranking framework, as one baseline. Experiments show that our approach can outperform baselines by up to 2.5% in NDCG score [16] (a metric for computing accuracy in prioritizing issues) and 214% in Pearson correlation coefficient [59] (a metric for computing correlations between two variables).

To measure the performance costs incurred by ads practically, we collect usage traces of 17 volunteer users for 20 Android apps containing ads. We focus on measuring four performance cost types: memory consumption, CPU utilization, network usage, and battery drainage, since these costs are commonly discussed in previous studies [80, 31]. The recorded usage traces were then replayed multiple times for simulating real usage scenarios and accurate cost measurement, resulting in the collection of more than 2,000 measurements for those apps. We measure the performance costs of ads based on these measurements.

We focus on answering the following questions:

**RQ1.** Do the performance costs of in-app ads significantly increase the no-ads versions? We re-analyze some of the questions (e.g., what is the energy cost of ads?) investigated by Gui *et al.* [31] by using practical usage traces for each subject app, whereas Gui *et al.* [31] use one experimental usage trace per app. This allows us to answer this question in a more realistic scenario.

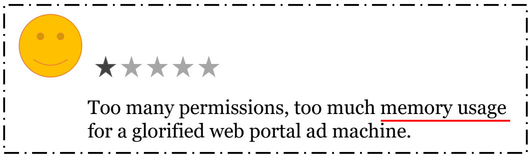
**RQ2.** How can the performance costs of ads affect user opinions? Based on the measured performance costs of ads, we empirically analyze the correlations between the costs and the user concerns quantified by RankMiner. We aim at exploring whether users pay more attention to performance costs.

The key *contributions* of this paper are as follows:

(a) We revisit some questions posed in previous work [31] by using practical usage traces. We find that performance costs of with-ads versions are significantly larger than those of no-ads versions with negligible effect sizes.

(b) We carry out the first empirical study to explore correlations between the performance costs of ads and their impact on user opinions, from which we deduce which cost types users care more about. We find that users are more concerned about the battery costs of ads, and tend to be insensitive to ads' data traffic costs.

(c) We make the source code[1] of the tools used to measure performance cost and to perform user review analysis publicly available to allow for replication and extension of our work.



★☆☆☆☆

Too many permissions, too much memory usage for a glorified web portal ad machine.

Figure 1: Example of user reviews. The red underline highlights one 2-gram term ("memory usage").

**Paper structure.** Section 2 describes the background and motivation of our work. Section 3 presents the methodology we propose for quantifying user concerns about specific app issues. Section 4 describes the results of our study. Section 5 discusses its limitation. Related work and final remarks are discussed in Section 6 and Section 7, respectively.

## 2. Background

In this section, we explain the concept of user reviews, the mobile advertising profit model, and the word embedding technique we utilize in app issue ranking.

### 2.1. User Reviews

User reviews on app distribution platforms, *e.g.*, Google Play, are posted by users to express their experience with apps. They generally serve as the primary channel for customers to leave feedback. As observed [7], two thirds of users leave reviews after negative experiences. The reviews, which usually report bugs, feature requests, and functionality improvement, provide valuable information to developers. Figure 1 depicts a review of an app publicly available from Google Play (The user's name and the date of the comment have been removed to preserve privacy). The user review complains about memory issues, indicated by the term "*memory usage*". Such information can be exploited by developers to discover user experience and improve app design accordingly. More importantly, reviews reflect real and immediate user response after interacting with apps, which cannot be easily collected by surveys. Thus, we leverage app reviews to capture user perceptions of in-app ads in this paper.

### 2.2. The In-App Advertising Ecosystem

The ecosystem for in-app advertising consists of four major ingredients, *i.e.*, app developers, advertisers, ad networks, and end users, as shown in Figure 2. To render advertising contents into an app, developers typically utilize third-party mobile ad SDKs which are provided by ad networks, such as AdMob [3] and MoPub [49]. The ad networks grant developers with options for ad display, *e.g.*, defining ad sizes. When loading a page embedded with ads, the app sends a request to the ad network for retrieving an ad. Finally, the fetched ad content is rendered on the user's screen. Developers can then get payment from advertisers according to the counts of ads displayed (*i.e.*, impressions) and clicked.

[1] https://remine-lab.github.io/adbetter.html

Figure 2: The in-app advertising ecosystem.

Users play an essential role in the ad-profiting process, since the number of ads displayed to users determines the ad revenue: User retention and a large user base are critical for app developers. However, embedding ads inappropriately can ruin user experience. According to a survey [5], two in three app users consider mobile ads annoying and tend to uninstall those apps or score them lower to convey their bad experience. Such negative feedback likely influences other potential users, which further leads to customer churn. Hence, exploring the effects of in-app ads on user experience is important for app monetization.

### 2.3. Word Embedding Techniques

Word embedding also known as word distributed representation [47, 77] is a technique for learning vector representations of words by training on a text corpus. Word embedding represents words as fixed-length real-valued vectors so that semantically or syntactically similar words are close to each other in the continuous vector space [47]. Word embeddings can be learnt using neural models such as Continuous Bag-of-Words (CBOW) or Skip-Gram [47], where the context words within a sliding window are involved during the learning process. Compared with traditional Bag-of-words approaches, e.g., counting word frequencies, word embeddings are low-dimensional (often tens or hundreds of dimensions), and thereby do not suffer from sparsity and inefficiency problem.

Likewise, a phrase (i.e., a term with more than one word) can also be embedded as a real-valued vector [88]. A basic way of phrase embedding is to view it as a bag-of-words and add up all its word vectors.

## 3. RankMiner: App Issue Ranking

Figure 3 shows the workflow of the proposed approach for quantifying user concerns about specific app issues, which mainly involves four steps: Review preprocessing, phrase retrieval, keyword extraction, and issue grading. We identify phrase candidates from preprocessed reviews, and extract keywords (including phrases and single words) related to specific app issue. Based on the related keyword list, we compute the user concern score about the issue. We elaborate on each step in the following.

Figure 3: Overview of RankMiner.

### 3.1. Preprocessing

App reviews are usually short in length and contain many casual words. To facilitate subsequent analysis, we eliminate the noisy characters in this step. We first convert all words into lowercase, and remove all non-English characters and non-alpha-numeric symbols. We retain the punctuations to ensure semantic integrity. Then, we reduce the words to their root forms by lemmatization [53] (e.g., was to be). Finally, we keep reviews with the number of words larger than three. We do not remove stop words [54] here for phrase retrieval in the next step. Since app reviews contain growing compound words (e.g., redownload), new words (e.g., galaxys8), and misspelled words (e.g., updte [update]), we do not involve the preprocessing methods in [82] where the custom dictionary may introduce information loss (e.g., over correction) in our situation.

### 3.2. Phrase retrieval

Phrase retrieval aims to identify the key terms (particularly those with multiple words) that are commonly used by users to voice their experience. The phrases are extracted here since one single word may be ambiguous in its semantic meanings without the context information. For example, in Figure 1, using either change or storage alone cannot reflect the comment completely, whereas the three consecutive words permit change storage can describe the user's viewpoint more accurately.

However, given that user reviews are casually written, extracting the meaningful phrases poses a challenge. In this paper, we adopt the typical Point-wise Mutual Information (PMI) method [39]. The PMI method measures the co-occurrence probabilities of two words, and thereby eliminating terms which are rarely used. The phrases we retrieve contain 2-gram terms (i.e., two consecutive words) and 3-gram terms (i.e., three consecutive words). Since phrases with more than three words rarely exist in the review collection, they are not extracted here. Equation (1) defines the PMI between two words $w_1$ and $w_2$:

$$\mathrm{PMI}(w_1; w_2) = \log \frac{Pr(w_1\ w_2)}{Pr(w_1)Pr(w_2)}, \qquad (1)$$

where $Pr(w_1\ w_2)$ and $Pr(w_i)$ denote the occurrence probabilities of the phrase $(w_1\ w_2)$ and the single word $w_i$, respectively. The terms with higher PMIs indicate that they appear together more frequently and tend to be semantically meaningful. The PMI thresholds are experimentally set. Based on the PMI results, we also ensure that at least one noun is included in each phrase via the Part-Of-Speech tagging method [73].

Table 1
Example of SentiStrength scores and defined sentiment scores for example review sentences.

| Review Sentence | SentiStrength Score | Defined Sentiment Score |
|---|---|---|
| Great but why make a browser if you don't have the resource to keep it up to date? | [3,-1] | 3 |
| Would be 5 stars if i could pay and remove all the ads. | [1,-1] | -1 |
| I like what it does but the additional stuff is annoying, eg loud video advert is disturbing. | [2,-3] | -3 |

### 3.3. Keyword Extraction

We propose to use an effective word representation approach, i.e., word emebdding [46] (introduced in Section 2.3), to discover semantically similar words and phrases for each app issue, where the app issue is usually described in keyword, e.g., privacy and crash.

We retrieve $k$ terms (including single words and phrases) most close to specific app issues based on the cosine distance of their vector representations, where $k$ is usually defined in the range of tens to hundreds. Due to the small number of the retrieved terms and also to ensure the keyword retrieval accuracy, we then manually trim noise words and phrases. The remaining terms are issue-related terms.

### 3.4. Issue Grading

We regard an review related to an app issue if the review containing any terms belonging to the issue-related terms. Similar to previous work [12, 84], we assume that issues complained in more reviews and yielding poorer ratings indicate higher concern levels among users, and need to be ranked higher. The time information (used by Chen et. al [12]) of the issues is not considered here, since we do not care about whether one issue is fresher than the others. In the following, we introduce the sentiment score and frequency score we adopt for grading app issues $\{I_1, I_2, \dots, I_i, \dots, I_w\}$ where $w$ is the number of app issue to be ranked.

Sentiment Score: The ratings provided by users may not be consistent with the sentiment expressed by their reviews. For example, one user describes Bad app in his/her review but gives a five-star rating. To mitigate this problem, we try to predict the actual sentiment of each user review. Inspired by Guzman and Maalej's work [33], we use SentiStrengh [75], a lexical sentiment extraction tool specialized in dealing with short, low-quality texts, for the sentiment analysis.

We first chunk the reviews into sentences by utilizing NLTK's punkt tokenizer [55]. Then we adopt SentiStrength to assign a positive and negative value to each review sentence, with positive scores in the range of [+1, +5], where +5 denotes an extremely positive sentiment and +1 denotes the absence of sentiment. Similarly, negative sentiments with the range [-5, -1], where -5 denotes an extremely negative sentiment and -1 indicates the absence of any negative sentiment. Table 1 displays examples of SentiStrength scores for review sentences. If the negative score multiplied by 1.5 is larger than the positive score, the sentence is assigned with the negative score. Otherwise, the final sentiment score is defined as the positive score. As explained in [33], multiplying the negative scores by 1.5 is considered due to the fact that users tend to write positive reviews [36]. We finally compute the sentiment score $R_i$ of the issue $I_i$ as the average sentiment scores of all its review sentences.

Frequency Score: The number of the reviews for issue type $I_i$ can be easily calculated, denoted as $N_i$.

Final Concern Score: The final concern score $U_i$ is defined in Equation (2), by combining the sentiment score $R_i$ and frequency score $N_i$.

$$U_i = -\log f(R_i) \cdot P_i , \tag{2}$$

where $P_i = N_i / N$, representing the percentage of the $I_i$-related reviews in the whole ad-related reviews. The function $f(R_i)$ is to confine the rating $R_i$ to be in the range $(0, 1)$, which is empirically defined as the soft division function, i.e., $(R_i - 0.9)/5$, or the sigmod function, i.e., $1/(1 + e^{-R_i})$. Equation (2) shows that issues with lower user ratings and larger review percentages will be given higher user concern values, which is consistent with our initial assumption.

## 4. Experimental Study

In this section, we first verify the effectiveness of RankMiner as a proof of concept, and then answer the following research questions as outlined in the Introduction:

RQ1. Do the performance costs of in-app ads significantly increase the no-ads versions?

RQ2. How can the performance costs of ads affect user opinions?

### 4.1. Proof of Concept: What is the accuracy of the proposed RankMiner approach?

#### 4.1.1. Motivation

By answering this question, we aim at verifying the effectiveness of RankMiner in quantifying user concerns about specific app issues. In this way, we can effectively measure user opinions about the performance costs of in-app ads based on RankMiner.

#### 4.1.2. Methodology

We conduct evaluation of our proposed strategy based on the reviews of Spotify Music provided by CrossMiner [42].

We employ this dataset due to its large number of app reviews and available ground truth (i.e., the official user forum [72]). The reviews are collected from three platforms: Android (178,477 reviews), iOS (249,212 reviews), and Windows Phone (33,143 reviews). The ground truth is defined based on the number of search results for each cost type on the user forum. We use this method to establish ground truth as an issue with more search results implies that the issue is encountered by more users, and thus collectively users care about it more. Pearson correlation coefficient (PCC) [59] is utilized to evaluate the linear correlation between measured user concerns and the numbers of user views for the four performance cost types. The typical metric NDCG@k (Normalized Discounted Cumulative Gain) [52] is adopted for computing the accuracy in prioritizing issues, i.e.,

$$
\begin{aligned}
NDCG@k &= \frac{DCG@k}{IDCG@k}, \\
\text{where } DCG@k &= \sum_{i=1}^{k} \frac{rank(i)}{\log_2(i+1)}, \\
\text{and } IDCG@k &= \sum_{i=1}^{rank_k} \frac{rank(i)}{\log_2(i+1)},
\end{aligned}
\tag{3}
$$

where $rank(i)$ indicates the ranking score of the $i$-th issue computed by Equ. (2) and $rank_k$ in computing $IDCG@k$ represents the position list based on the computed ranking scores. The premise of DCG is that highly important issues appearing lower in the prioritized results should be penalized as the ranking score is reduced logarithmically proportional to the position of the issues. IDCG (Ideal DCG) computes the maximum DCG based on the position list resulted by the ranking scores. $NDCG@k \in [0,1]$, and $k$ denoting the number of elements to be sorted. Higher NDCG values represent that DCG values are close to the IDCG results, implying more accurate rankings. NDCG@k is computed by comparing the rank of the measured user concerns for the four permanence cost types (e.g., CPU cost) with the rank of the user view numbers in the ground truth.

We measure users' concerns about the performance costs, including memory, CPU, network, and battery, of the Spotify Music apps based on RankMiner. Specifically, we capture top 50 (i.e., $k$=50) terms that are semantically close to the target cost type, e.g., battery. We further manually remove ambiguous and noisy ones from the captured top terms, such as the terms data volume and data plan, shown in the box below. The remaining terms are battery-related terms. We also notice that misspelled words (e.g., batery) can be captured through word embeddings. Table 2 illustrates the terms related to each performance cost types.

Battery-related terms: battery life, ~~data volume~~, batery, battery power, ~~data plan~~, battery juice, ...

Table 2
Performance-related terms.

| Cost Type | #Terms | Related Term |
|---|---|---|
| Memory | 19 | ram memory, storage, storage space, memory space, space, internal memory, ram, internal storage, internal space, disk space, gb, battery power, extra space, ram memory, unnecessary space, capacity, mb, valuable space, precious space |
| CPU | 17 | cpu, processor, gpu, cpu usage, laggy, slowly, too slow, incredibly slow, extremely slow, sluggish, painfully slow, terribly slow, take age, slower, slower than before, lag, fast |
| Network | 12 | network connection, data connection, wifi connection, network signal, wifi, wifi network, wifi signal, internet connectivity, wireless connection, 4g connection, internet connection, wireless network |
| Battery | 14 | battery life, battery power, batery, batt, battery drain, battery usage, battery rapidly, battery dry, battery overnight, battery juice, batterie, battery excessively, battery life, drain battery |

Table 3
Results of comparison with baselines. The subscripts beside the correlation coefficients indicate the corresponding resulted p-values.

| | CrossMiner (Percent-Based) | Rating-Based | RankMiner (Sigmod) | RankMiner (Soft Division) |
|---|---|---|---|---|
| PCC | $0.728_{0.272}$ | $0.253_{0.747}$ | $0.783_{0.218}$ | $0.794_{0.206}$ |
| NDCG@4 | 0.854 | 0.869 | 0.875 | 0.875 |

### 4.1.3. Results

Table 3 depicts the comparison results of our methods (sigmod and soft division methods) with two baseline methods, one is only based on the review percentage (i.e., the CrossMiner method [42]) and the other is based on the user sentiment $R_i$ in Equation 2). We validate the quantified users' complaints about the four types of costs (memory, CPU, network and battery). As Table 3 shows, our methods achieve the best properties than the basic methods in terms of both PCC and NDCG@4, where NDCG@4 measures the accuracy in ranking four types of costs. Specifically, the soft division and sigmod methods can better identify important performance issues, with an increase of 2.5% for NDCG@4 compared to CrossMiner [42]. For the PCC results, the soft division method surpasses the ratings-based method by 2.14 times in terms of the correlation coefficients, however the p-values ($> 0.05$) show that the correlations are not statistically significant, which means the relations between different ranking scores and the ground truth may be weak. This could be attributed to the small sample size involved. Overall, the proposed methods can prioritize the issues more accurately by balancing review ratings and percentages. During the analysis, we adopt the soft division method, which achieves the most optimal results in our comparison, for scoring users' concerns.

Finding 1: The proposed RankMiner approach can effectively quantify user concerns about specific app issues.

Table 4
Subject apps used to answer RQ2 in our empirical study.

| Category | ID | App Name | Package Name | Version | # Reviews | Overall Rating |
|---|---|---|---|---|---|---|
| Weather | A1 | RadarNow! | com.usnaviguide.radar_now | 6.3 | 2,346 | 4.4 |
| | A2 | Transparent clock & weather | com.droid27.transparentclockweather | 0.99.02.02 | 918 | 4.3 |
| | A3 | Weather Underground: Forecasts | com.wunderground.android.weather | 5.6 | 4,584 | 4.5 |
| | A4 | AccuWeather | com.accuweather.android | 4.6.0 | 8,691 | 4.3 |
| Productivity | A5 | QR & Barcode Scanner | com.gamma.scan | 1.373 | 297 | 4.3 |
| | A6 | Advanced Task Killer | com.rechild.advancedtaskkiller | 2.2.1B216 | 358 | 4.4 |
| | A7 | Super-Bright LED Flashlight | com.surpax.ledflashlight.panel | 1.1.4 | 1,661 | 4.6 |
| | A8 | iTranslate - Free Translator | at.nk.tools.iTranslate | 3.5.8 | 242 | 4.4 |
| | A9 | AVG Cleaner for Android phones | com.avg.cleaner | 3.7.0.1 | 494 | 4.3 |
| Health & Fitness | A10 | Pedometer | com.tayu.tau.pedometer | 5.19 | 2,024 | 4.4 |
| | A11 | Pedometer & Weight Loss Coach | cc.pacer.androidapp | 2.17.0 | 1,576 | 4.5 |
| | A12 | Period Tracker | com.period.tracker.lite | 2.4.4 | 1,332 | 4.5 |
| | A13 | Alarm Clock Plus? | com.vp.alarmClockPlusDock | 5.2 | 577 | 4.4 |
| | A14 | Daily Ab Workout FREE | com.tinymission.dailyabworkoutfree1 | 5.01 | 25 | 4.4 |
| | A15 | Map My Ride GPS Cycling Riding | com.mapmyride.android2 | 17.2.1 | 408 | 4.4 |
| | A16 | Calorie Counter - MyFitnessPal | com.myfitnesspal.android | 6.5.6 | 2,267 | 4.6 |
| News & Magazines | A17 | BBC News | bbc.mobile.news.ww | 4.0.0.80 | 9,693 | 4.3 |
| | A18 | Fox News | com.foxnews.android | 2.5.0 | 4,163 | 4.5 |
| | A19 | NYTimes - Latest News | com.nytimes.android | 6.09.1 | 71 | 3.8 |
| | A20 | Dailyhunt (Newshunt) News | com.eterno | 8.3.17 | 1,452 | 4.3 |

Figure 4: Workflow of performance costs of in-app ads.

## 4.2. RQ1: Do the performance costs of in-app ads significantly increase the no-ads versions?

### 4.2.1. Motivation

We revisit some of the questions (i.e., what is the energy/network/memory/CPU cost of ads?) investigated by Gui et al. [31]. Differently from previous work [31], which only uses one experimental usage trace per app, we collect practical usage traces of subject apps. We want to examine whether the performance costs of in-ads apps and their no-ads versions exist significant differences in a relatively more practical scenario.

### 4.2.2. Methodology

The workflow for measuring performance costs of in-app ads can be found in Figure 4.

Subject App Selection.We select 20 popular apps from Google Play as the subjects based on the following four criteria: (1) they are selected from different categories - to ensure the generalization of our results; (2) they are apps containing ads; (3) they have a large number of reviews - indicating that user feedback can be sufficiently reflected in the reviews; and (4) they can be convertible to no-ads versions - for measuring the costs caused by ads. To collect apps that satisfy the first criterion, we randomly search the top 20 apps in each of the categories (except games and family apps) on Google Play. Since Google Play provides the number of reviews and declaration about ads, we extract apps with more than 10,000 reviews and with ads contained. To satisfy the last criterion, we convert these apps to no-ads versions based on Xposed[86] in a random order and then inspect whether the ads had been successfully removed. Finally, we choose 20 subjects for our experiment analysis. Their details are illustrated in Table 4, where we list the category, app name, package name, version, number of reviews, and overall rating for each subject app.

Usage Trace Collection. For rendering the viewing traces of ads various, 17 users are selected from different genders (six females and 11 males), and distributed in different age groups (six of them are aged at 18-25, ten at 25-30, and one at 30-35). All the selected participants satisfy the following criteria: 1) they interact with apps for more than 30 minutes daily - indicating that the users are familiar with using mobile apps; 2) they have experience using apps of different categories - considering the multi-categories of the subject apps; and 3) they are willing to spend time on our experiments - implying that they will take patience to execute the apps according to their usual habits We invite them to exercise the functionalities of the 20 subject apps according to their own usage habits.

Table 5 depicts the statistics of the duration for the collected user traces, including the maximum, minimum, and average duration for each app. We can observe that the average interaction time for the apps ranges from 14 seconds to 2.48 minutes. Short interaction spans may be attributed to the simple functionality provided by some apps. For exam-

Table 5
Statistics of duration for collected usage traces.

| ID | Max. (s) | Min. (s) | Avg. (s) |
|---|---|---|---|
| A1 | 155.12 | 66.36 | 19.89 |
| A2 | 92.76 | 25.37 | 61.18 |
| A3 | 125.80 | 20.28 | 67.93 |
| A4 | 153.56 | 24.30 | 68.44 |
| A5 | 42.77 | 0.07 | 14.51 |
| A6 | 59.34 | 3.01 | 23.49 |
| A7 | 69.36 | 4.48 | 23.50 |
| A8 | 167.59 | 28.37 | 65.30 |
| A9 | 331.12 | 56.34 | 13.24 |
| A10 | 143.03 | 11.34 | 58.35 |
| A11 | 153.71 | 11.34 | 64.44 |
| A12 | 154.18 | 33.79 | 96.72 |
| A13 | 134.44 | 20.98 | 69.79 |
| A14 | 210.15 | 25.74 | 105.03 |
| A15 | 230.95 | 22.46 | 102.43 |
| A16 | 501.06 | 13.57 | 149.52 |
| A17 | 325.52 | 15.10 | 96.64 |
| A18 | 292.97 | 6.88 | 88.64 |
| A19 | 243.32 | 27.35 | 100.29 |
| A20 | 190.19 | 11.92 | 87.72 |

ple, the app com.rechild.advancedtaskkiller (A6) mainly supports service killing by clicking one button on the home page, which costs about 23 seconds on average according to our records. At least 70% apps are executed for more than one minute on average, and only one app com.gamma.scan (A5) is executed with less than 20 seconds.

For each app, we measure 102 times[2] by repeating both the with-ads version and the no-ads version three times using RERAN [26]. Whether the differences of the collected statistics for the 102 runs on each app are significant or not is not examined. The average values are calculated to alleviate noises. To mitigate background noise, we restore the system environment to its original state before each version execution. Then we install the app and start its execution. When a subject app is launched, the tools tcpdump and top are started to capture the transmitted data traffic and memory/CPU consumption, respectively. We also monitor the app execution to ensure that they are consistent with the records. Note that even though running tools, such as top and Xposed, can affect mobile performance, the effects could be consistent on both versions (with-ads and no-ads) [31] and can thus be ignored in our cost measurement. Overall, we measure the 20 subject apps 2,040 times[3] in total. The whole measurement process lasted for for more than one month.

Performance Cost Measurement. We measure the memory, CPU and network costs following Gui et al. [31], and battery cost following Gao et al. [24]. The ad costs are computed by subtracting the costs of no-ads versions from those of with-ads versions.

Table 6
Average and standard deviation of the increase rate of performance cost when comparing with-ads version with the no-ads version.

| Cost Type | Memory | CPU | Network | Battery |
|---|---|---|---|---|
| Average | 25.2% | 6.9% | 113.9% | 17.7% |
| Standard Deviation | 12.5% | 3.7% | 108.9% | 11.9% |

### 4.2.3. Results

For each subject app, we measure the four types of performance costs (*i.e.*, memory, CPU, network and battery consumption) for both with-ads and no-ads versions. Figure 5 depicts the costs of the 20 apps, with blue bars indicating the memory costs of the no-ads versions and orange bars representing the ad costs. According to the figure, all the with-ads versions consume more performance cost than their no-ads versions. For example, with ads integrated, the CPU cost of A10 has apparently increased, and the network usage of subjects such as A6 and A12 shows dramatic growth. The memory increase ranges from 5.9% (A16) to 46.4% (A6), with an average of 25.2%. For CPU cost, ads in the subject apps consume 1.0% to 12.0% with respect to the CPU occupation rate, with median cost at 7.4%. This indicates that mobile ads indeed influence the device resource, which is consistent with the results by Gui *et al.* [31].

Table 6 shows the statistics of all measured performance costs for the 20 subjects, with the average increase rate and corresponding deviation (which represents the cost increase variations among the subject apps). Network usage has the most remarkable increase (113.9%) on average. The distinct cost increase (s.d. at 108.9%) of network usage may be attributed to the ads-oriented design of some apps. CPU costs experience a modest increase (6.9% on average). Moreover, the growth in battery drainage is also noteworthy, with the average increase at 17.7% and deviation at 11.9%. Heavy performance costs may ruin user experience and drive users to uninstall the apps, which is the reason why developers and researchers pay attention to ad performance costs.

We further observe whether statistically significant differences exist between performance costs of with-ads versions and those of no-ads versions. We first check the distributions of each type of measured performance costs by the Shapiro-Wilk test [69]. The Shapiro-Wilk test is a typical test of normality of which the null hypothesis is that the input samples come from a normally distributed population. If the p-value computed by the Shapiro-Wilk test is smaller than 0.05, we achieve that the input distribution is significantly different from normal distribution. Table 7 lists the p-value results of Shapiro-Wilk test for different performance cost types. We observe that except for the traffic cost of with-ads versions, all the other measured costs render normal distributions. This may be because traffic is more sensitive to the usage pattern and time of various users. Therefore, for memory, CPU and battery costs, we use the paired t-test [34] for comparing the distributions between with-ads and no-ads versions, and use the Wilcoxon signed-rank test for analyzing the traffic costs. The paired t-test is a statistical test to

[2] 102 = 17 · 6, where 17 is the number of volunteer users and six denotes the total measuring times for both the with-ads and no-ads versions of an app.

[3] 2040 = 102 · 20, where 20 denotes the number of subject apps.

(a) Memory

(b) CPU

(c) Battery

(d) Traffic

Figure 5: RQ3: Performance consumption of with-ads (in orange) and no-ads versions (in light blue).

Table 7
Normality test of differences between measured performance costs of with-ads versions and no-ads versions. The p-value $< 0.05$ means the differences are not normally distributed.

| Cost Type | Memory | CPU | Battery | Traffic |
|---|---|---|---|---|
| p-value | 0.666 | 0.116 | 0.429 | 0.001 |

determine whether the mean difference between paired observations is zero, with the p-value less than 0.05 indicating the difference between two paired inputs is significant. We use paired t-test for costs of memory, CPU, and Battery, because the subject apps may have different cost values for with-ads and no-ads versions, and the differences between pairs are normally distributed. The Wilcoxon signed-rank test is a paired version of the Wilcoxon rank-sum test.

Figure 6 illustrates the comparison on the performance costs of with-ads and no-ads versions. The p-values in paired t-test and Wilcoxon signed-rank tests show that the two input distributions are significantly different. The effect sizes measured by Vargha and Delaney's $A_{12}$ are all negligible. The results indicate that versions with ads expend significantly more performance costs, which is consistent with the studies in [31] and [67].

Finding 2: Performance costs of with-ads versions are significantly larger than those of no-ads versions.

## 4.3. RQ2: How can the performance costs of ads affect user opinions?

### 4.3.1. Motivation

We aim at exploring whether users show more concerns for more performance costs of in-app ads, and which performance cost users care more about. Thus, developers can understand more about user perceptions of in-app ads, and pay more attention to user-concerned performance costs.

### 4.3.2. Methodology

We crawl totally 34,455 reviews published from December, 2016 to April, 2017 for the 20 apps. The reviews are large enough for review analysis [12], which can effectively capture the user experience. To ensure that user reviews are specific to subject app versions, we select the reviews posted by users within two months[4] after the corresponding version release.

We first retrieve ad-related reviews by extracting the reviews explicitly related to ads, i.e., reviews containing words such as ad, ads, or advert* (with regular expression) [31]. Then we measure users' concerns about the performance costs, including memory, CPU, network, and battery, of both the with-ads apps and in-app ads based on RankMiner. We calculate user concerns about ads' performance costs based on the ad-related reviews only.

### 4.3.3. Results

We illustrate the results of users' concerns about the performance costs in Figure 7, with the blue bars and orange bars denoting the measured values for no-ads and with-ads versions respectively. For the 20 subjects, users express different levels of concerns about the memory overhead of the in-app ads. For example, for the memory cost, A2 receives the most complaints about ads among all the subject apps, with an obvious increase of 35.9% compared with the no-ads version indicated by blue bar in Figure 7. By inspecting A2, we discover that in-app ads can occupy almost the whole screen space, especially with one banner on the top and one rectangle ad appearing in the middle when sliding downward. Interestingly, we find that 15 (75%) apps receive zero negative feedback about the memory costs of ads (i.e., only blue bar is shown for the app in Figure 7), such as A1. This implies that in most cases, user tend to be insensitive to

[4] The period is defined following previous work [15].

(a) Memory Cost (b) CPU Cost (c) Battery Cost (d) Traffic Cost

Figure 6: Performance cost distributions for with-ads (in purple) and no-ads versions (in light blue).

(a) Memory Cost (b) CPU Cost

(c) Battery Cost (d) Traffic Cost

Figure 7: Quantified user concerns about different performance cost types of the 20 subject apps.

the memory costs caused by in-app ads.

By observing the increase rate of quantified user concerns about all performance costs (shown in Table 8), we identify that memory costs have the largest rate of growth in user concerns (6.3% on average) and the most obvious deviation (17.0%) among the 20 apps. However, users express the least concerns about network costs, with the increase rate averaging at 0.9% and a deviation of 1.9%. Such an observation is different from what we have discovered in Table 6, where network costs exhibit the highest increase among all the performance costs. We find that 15/20, 12/20, 15/20, and 15/20 of the subject apps do not receive any complaints from users regarding the cost of memory, CPU, battery, and traffic, respectively. We think that users may perceive different types of performance costs differently. We then conduct correlation analysis to explore that there are strong correlations between user concerns and performance costs of ads. Specifically, we use PCC to calculate the correlations between the quantified user concerns and measured costs of the 20 subjects for each performance cost type.

The correlations between performance costs and the corresponding user concerns are illustrated in Table 9. Almost all the PCC results indicate that their linear correlations are weak, especially for memory usage which represents nearly no correlation with the quantified user concerns (with PCC score $r_p = -0.132$). The only one performance type that presents moderate correlation with the quantified user concern is battery cost, with $r_p = 0.534$ and $p = 0.015 < 0.05$.

Table 8
Increase rate of quantified user concerns about performance costs.

| Cost Type | Memory | CPU | Network | Battery |
|---|---|---|---|---|
| Average | 6.3% | 3.5% | 0.9% | 2.7% |
| Standard Deviation | 17.0% | 8.9% | 1.9% | 9.6% |

The results of PCC are consistent with those of SRC, where user concern shows a strongly increasing trend with more battery consumed ($p = 0.0009 < 0.05$). This allows us to conclude that users care most about the battery cost among all the performance cost types. We attribute this to that the consumption of battery is more sensible than other cost to users, and therefore more battery costs tend to cause more unfavorable reviews.

We also observe the negative correlation between network cost and the corresponding user concern with respect

Table 9
Correlation test results between performance costs of ads and user concerns.

| Cost Type | Memory | | CPU | | Network | | Battery | |
|---|---|---|---|---|---|---|---|---|
| | r-score | p-value[3] | r-score | p-value | r-score | p-value | r-score | p-value |
| PCC[1] | 0.132 | 0.578 | 0.166 | 0.482 | -0.281 | 0.229 | 0.534 | 0.015 |
| SRC[2] | 0.372 | 0.105 | 0.213 | 0.366 | -0.127 | 0.591 | 0.679 | 0.0009 |

[1,2] The absolute values of the PCC/SRC scores represent very weak correlations if $|r| < 0.2$, weak correlation if $0.2 \le |r| < 0.4$, moderate correlations if $0.4 \le |r| < 0.6$, strong correlations if $0.6 \le |r| < 0.8$, and very strong correlation if $|r| \ge 0.8$ [19].

[3] $p < 0.05$ indicates that the correlation is statistically significant.

to both PCC and SRC analysis. This means that more network costs could possibly bring better user experience. This might be against our common sense. We attribute this to the ubiquity of WiFi leading to fewer concerns about traffic consumed. According to [79], over 90% of users choose WiFi connections when using smartphones. We therefore conclude that the network consumption of ads may not be concerned to users.

For CPU costs, the PCC ($r_p = 0.166$) and SRC ($r_s = 0.213$) scores display weak correlations with user concerns. The result is predictable, as users may not perceive the CPU cost on their mobile phones, and would generally think the crash or laggy performance is caused by mobile systems or app-specific functionalities. We conclude that the effect of CPU consumption on users is weak. Note that since our data are not time-series, causal impact analysis [43, 9] is not applicable in our situation. Moreover, our correlation analysis is applicable and convincing to determine the correlations between the two factors.

Finding 3: Users care most about the battery cost among all the performance cost types, and show least sensitivity to the data traffic cost of ads.

## 5. Discussion and Limitation

In this section, we discuss the threats to the validity of our study and illustrate the steps we have taken to mitigate them. We also discuss the usefulness of our findings.

### 5.1. Threats to validity

External Validity: First, our experimental study is based on limited real apps from Google Play. Although the study does not involve other app distribution platforms (e.g., App Store), we believe our results would also work across the board, since ad rendering mechanisms are similar across app markets. We determine the number of subjects following prior work [31], which achieves the finding that apps with ads have more hidden cost than those without ads based on 20 subject apps. In this paper, we alleviate this threat by ensuring that the subject apps are popular apps distributed in four different categories. We argue that future replications with similar contexts, e.g., using similar apps created in similar organizations, are likely to achieve identical observations as ours. Future work will consider more apps and app platforms. Second, we collect usage traces from 17 volunteers which account for a limited number of the whole audience. In our experiments, ad displaying periods can impact the measured performance costs of ads. Since ads are generally set to refresh about every 60 seconds [63], the collected usage traces would cover different situations of ad rendering and reloads (with the minimum interaction spans range from 0.06s to 33.8s and the maximum from 42.8s to 8.4min for the apps). Moreover, we invite the volunteers from different age groups and genders, which enriches the usage traces of in-app ads. Besides, there are no available datasets about the performance costs of ads or systematic tools for collecting all the performance consumed by ads on a large scale. Our work is the first to explore the performance costs of ads in practice.

Internal Validity: First, we leverage the Xposed and AdBlocker Reborn modules for generating no-ads versions, which may introduce additional workload to mobile apps. Since we instrument Xposed, which has been widely used in performance testing and bug detection [37, 10], into the phones for both with-ads and no-ads versions, the influence of Xposed is consistent and can be eliminated by subtracting the costs of the two versions. We then just verify the influence of AdBlocker Reborn on mobile performance. The costs are measured for three apps (including MediCalc, Google Maps, and RealCalc Plus) with the module enabled and disabled, respectively. The results exhibit that the average increase rates in costs are 3.0%, 0.6%, and 0.0% for the memory, CPU, and battery, respectively. Compared with the performance consumption of each subject app, such cost increase is negligible.

Second, user concerns are quantified based on the proposed RankMiner and user reviews, which might not represent the real opinions of some users. To verify the effectiveness of RankMiner, we compare with baselines and our soft-division based method shows significant increase in accuracy (e.g., 214% increase compared to the rating-based method in PCC). Besides, Google Play does not provide access to all the user reviews. Hence, any analysis on the user reviews might encounter dealing with an incomplete set of data [56]. To reduce such a bias on the findings, we collect all the reviews from December 2016 to April 2017 for the subject apps (1,722 reviews on average).

Third, the focus of our study is to examine the association between performance costs of ads and the user concerns. Note that association does not imply causation. Furthermore, we do not have records of consecutive app versions, we do lot have a complete picture. This limitation is shared by previous studies [66, 8, 76] that analyze relationships between app characteristics and user ratings. Still, these studies along with ours are the first steps towards understanding of the factors that impact user ratings/concerns. In future, more advanced statistical analysis, e.g., causality analysis [64], can also be employed.

Finally, to alleviate background noise and obtain reliable performance cost values, we measure 51 times for each app version and a total of more than 2,000 times for all the subject apps. The average results are utilized for our study.

### 5.2. Usefulness of our findings

We adopt the Technology Acceptance Model (TAM) [17], the most influential models of technology acceptance [11], to analyze the usefulness of our findings. TAM summarizes two primary factors that can influence an individual's intention to adopt a technology: perceived ease of use (PEOU) and perceived usefulness (PU). Based on TAM [17], the PEOU factor in our scenario could be affected by the developers' experience and voluntariness. We suppose that the developers are experienced in in-app ad design and voluntary to apply our findings to their practical development; so the PEOU factor is favorable for the usage of our findings. For the PE factor, it could be impacted by developers' subjective norm, their understanding of our findings, job relevance, expectation of higher quality, and demonstrability of the results, besides the PEOU factor. In the study, we have demonstrated that the obvious performance costs of in-app ads versions and the users' sensitivity to the performance costs through practical experiments, based on which we suppose that the developers believe our findings are meaningful and comprehend them well. We also assume that the developers do not refuse to try the findings to mitigate the performance costs of the in-app ads. The expected results can be better user experience or app revenue. Therefore, the PE factor would also be positive for the adoption of our findings, and the developers will have the attitude and intention to use the findings. However, the perception may change depending on age and gender [17].

### 5.3. Common ad-related terms in ad reviews

To take a deep look into what users commonly complain about ads, we use RankMiner to identify ad-related terms and quantify user concern of each term. The ad-related terms are determined by retrieving most similar terms to ad or ads following the method in Section 3.3. We find that users mentioned most about ad content (e.g., spam), appearance style (pop up ad), ad size (e.g, full screen ad), ad timing (e.g, 30 second ads), and obstruction (e.g., intrusive ad). We manually label the ad-related terms into these five groups, and visualize them for readers to better understand the extracted common ad-related complaints. We can discover that users are concerned about various aspects of advertising in apps besides the performance costs studied in this paper. Future research can extend our research by analyzing user perceptions of these aspects.

### 5.4. Implications of our study

For practitioners: The finding that performance costs of in-app ads versions are significantly larger than those of no-ads versions indicates that practitioners should notice the performance costs of in-app ads. The finding that users care most about the battery cost among all the performance cost types, suggests that practitioners should focus on the battery cost of in-app ads instead of treating all the cost types equally. To alleviate the negative impact of battery cost, practitioners can conduct A/B testing experiments to measure the battery cost of the in-app ads with different rendering strategies, e.g., different video resolutions and image sizes.

Figure 8: Visualization of ad issues. Larger bubbles indicate that the corresponding terms are of more concern to users.

For researchers: More research on mitigating the costs of in-app ads including other hidden costs, such as app maintenance effort caused by in-app ads, is encouraged. Although anecdotal evidence exhibits the hidden costs of in-app ads, few research work has explored how to properly design mobile ads to mitigate the costs while preserving user experience (e.g., which rendering modes, such as image/video, of in-app ads are more favorable).

## 6. Related Work

We present two lines of work that inspire our study on in-app ads: app review analysis and ad cost exploration. A comprehensive survey on app store analysis for software engineering can be found elsewhere [44].

### 6.1. App review analysis

App review analysis explores the rich interplay between app customers and their developers. App reviews are a valuable resource provided directly by the users, which can be exploited by app developers during bug-fixing [4, 60] and feature-improving process [22]. In previous work [36], the authors manually label 3,278 reviews of 161 apps, and discover the most recurring issues users report through reviews. Since mining app reviews manually is labor-intensive due to the large volume, more attempts on automatically extracting app features are conducted in prior studies. For example, Iacob and Harrison [35] design MARA for retrieving app feature requests based on linguistic rules. Ma et al. [42] propose a word2vec-based approach for collecting descriptive words for specific features, where word2vec [47] is utilized to compute semantic similarity between two words. Another line of work focuses on condensing feature information from reviews and captures user needs to assist developers in performing app maintenance [18, 81]. There are also investigations aiming at extracting valuable information from user reviews for supporting the evolution of mobile apps [23, 58].

Previous research [82, 83] has also investigated how to facilitate keyword retrieval and anomaly keyword identification by clustering semantically similar words or phrases.

Other work [33, 29, 40] propose methods to identify users' opinions about specific app features/aspects. Detailed literature about opinion mining from app reviews can be found in the work by [25]. We use the sentiment prediction method proposed by Guzman et al. [33] for computing the sentiment score in RankMiner. Besides, the keyword extraction step in RankMiner builds on the work of Vu et al. [82] by extending the keyword lists with phrases instead of using single words only.

### 6.2. Ad cost exploration

Mobile ads can generate several types of costs for end users, e.g., battery drainage [50], privacy leakage [13, 62, 45], and traffic data cost [61]. According to the research [38], privacy & ethics and hidden cost are the two most negatively perceived complaints (and are mostly in one-star reviews) among all studied complaint types. The work by Son et al. [70] shows that malicious ads can infer sensitive information about users by accessing external storage. Stevens et al. [74] investigate the effect on user privacy of popular Android ad providers by reviewing their use of permissions. The authors show that users can be tracked by a network sniffer across ad providers and by an ad provider across applications. The study by Gui et al. [30] proposes several lightweight statistical approaches for measuring and predicting ad related energy consumption, without requiring expensive infrastructure or developer effort. Wei et al. [85] and Nath et al. [51] discover that the free nature of apps comes with a noticeable cost by monitoring the traffic usage and system calls related to mobile ads. The work by Ullah et al. [78] finds that although user's information is collected, the subsequent usage of such information for ads is still low. Ruiz et al. [65] also explores how many ad libraries are commonly integrated into apps, and whether the number of ad libraries impacts app ratings. The authors find no evidence that the number of ad libraries in an app is related to its possible rating in the app store, but integrating certain ad libraries can negatively impact an app's rating.

To alleviate these threats, Mohan [48] and Vallina-Rodriguez et. al [80] develop a system to enable energy-efficient ad delivery. In the work of Seneviratne [68], the authors propose the architecture MASTAds allowing ad networks to obtain only the necessary information in providing targeted advertisements with user privacy preserved. An interesting empirical study by Gui [31] exhibits obvious hidden costs caused by ads from both developers' perspective (i.e., app release frequencies) and users' perspective (e.g., user ratings). Saborido [67] further highlight that ad-supported apps consume more resources than their corresponding paid versions with statistically significant differences. The work by Gao [24] investigates the performance costs raised by different advertisement schemes. In particular, they carried out an empirical study by considering 12 ad schemes from three different ads providers and analyzing three types of performance costs (memory/CPU, traffic and battery). The results of their study indicate that some ad schemes that produce less performance cost and provide suggestions to developers on ad scheme design.

In terms of performance cost measurement, the closest studies to our work are those by Gui [31] and Gao [24]. Different from them, we focus on analyzing the correlations between the performance costs of ads and users' attitudes. Besides, our performance costs are measured based on collected practical usage traces instead of experimental usage paths, which gives further confirmation on the findings by Gui [31].

## 7. Conclusion

In this paper, we have explored the effects of the performance costs of in-app ads on user experience. We propose an approach, named RankMiner, for quantifying user concerns about app issues. The usefulness of RankMiner is embodied in that it can be beneficial for product managers to assess users' attitude towards specific app features and app testers to pinpoint possible app bugs based on the quantified user concerns. Besides, the deployment of RankMiner requires no professional knowledge about the involved techniques, reflecting its feasibility in practical technology transfer. In this work, we adopt RankMiner to measure user opinions about the performance costs of ads. We find that performance costs of with-ads versions are significantly larger than those of no-ads versions with negligible effect sizes. By analyzing the correlations between the ads' performance costs and their impact on user opinions, we find the cost types that are more cared by users. We find that users are more concerned about the battery costs of ads, and tend to be insensitive to ads' data traffic costs. In future, we will extend our experiments by involving more apps, and study how to alleviate the battery costs when rendering ads.

## References

[1] Ad report, 2017. A hand-held world: the future of mobile advertising. http://www.business.com/mobile-marketing/the-future-of-mobile-advertising/ .

[2] Ad survey, 2016. Top 7 reasons why people uninstall mobile apps. http://cdn2.hubspot.net/hubfs/355159/10_Reasons_Why_Users_Uninstall_Your_Mobile_App.pdf .

[3] AdMob, 2018. AdMob. https://www.google.com/admob/ .

[4] Ali, N., Guéhéneuc, Y., Antoniol, G., 2013. Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. IEEE Trans. Software Eng. 39, 725 741.

[5] Annoying ad, 2016. Which ads do Internet users dislike the most? http://www.marketingcharts.com/online/which-ads-do-internet-users-dislike-the-most-69268/ .

[6] App market, 2018. Distribution of free and paid Android apps in the Google Play. https://www.statista.com/statistics/266211/distribution-of-free-and-paid-android-apps/ .

[7] App ratings, 2015. Import app ratings and reviews. https://pm.appsee.com/2015/12/07/how-important-are-app-ratings-and-reviews-to-users/ .

[8] Bavota, G., Vásquez, M.L., Bernal-Cárdenas, C.E., Penta, M.D., Oliveto, R., Poshyvanyk, D., 2015. The impact of API change- and fault-proneness on the user ratings of android apps. IEEE Trans. Software Eng. 41, 384 407.

[9] Brodersen, K.H., Gallusser, F., Koehler, J., Remy, N., Scott, S.L., et al., 2015. Inferring causal impact using bayesian structural time-series models. The Annals of Applied Statistics 9, 247 274.

[10] Casati, L., Visconti, A., 2017. Exploiting a bad user practice to retrieve data leakage on android password managers, in: Proceedings of International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Springer. pp. 952 958.

[11] Charness, N., Boot, W.R., 2016. Technology, gaming, and social networking, in: Handbook of the Psychology of Aging. Elsevier, pp. 389 407.

[12] Chen, N., Lin, J., Hoi, S.C., Xiao, X., Zhang, B., 2014a. Ar-miner: mining informative reviews for developers from mobile app marketplace, in: Proceedings of the 36th International Conference on Software Engineering (ICSE), ACM. pp. 767 778.

[13] Chen, T., Ullah, I., Kâafar, M.A., Boreli, R., 2014b. Information leakage through mobile analytics services, in: 15th Workshop on Mobile Computing Systems and Applications, HotMobile '14, Santa Barbara, CA, USA, February 26-27, 2014, pp. 15:1 15:6.

[14] Chowdhury, H.K., Parvin, N., Weitenberner, C., Becker, M., 2006. Consumer attitude toward mobile advertising in an emerging market: An empirical study. International Journal of Mobile Marketing 1.

[15] Ciurumelea, A., Schaufelbuhl, A., Panichella, S., Gall, H.C., 2017. Analyzing reviews and code of mobile apps for better release planning, in: IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER 2017, Klagenfurt, Austria, February 20-24, 2017, pp. 91 102.

[16] Croft, W.B., Metzler, D., Strohman, T., 2010. Search engines: Information retrieval in practice. volume 520. Addison-Wesley Reading.

[17] Davis, F.D., 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. 13, 319 340.

[18] Di Sorbo, A., Panichella, S., Alexandru, C.V., Shimagaki, J., Visaggio, C.A., Canfora, G., Gall, H.C., 2016. What would users change in my app? summarizing app reviews for recommending software changes, in: Proceedings of the 24th SIGSOFT International Symposium on Foundations of Software Engineering (FSE), ACM. pp. 499 510.

[19] Evans, J.D., 1996. Straightforward statistics for the behavioral sciences. Brooks/Cole.

[20] Facebook ads, 2017. Facebook ad revenue growth. https://bit.ly/2UlQRa4.

[21] Facebook reports, 2016. Facebook reports first quarter 2016 results. https://bit.ly/2p8EB9m .

[22] Gao, C., Wang, B., He, P., Zhu, J., Zhou, Y., Lyu, M.R., 2015. PAID: prioritizing app issues for developers by tracking user reviews over versions, in: ISSRE, IEEE Computer Society. pp. 35 45.

[23] Gao, C., Zeng, J., Lyu, M.R., King, I., 2018a. Online app review analysis for identifying emerging issues, in: ICSE, ACM. pp. 48 58.

[24] Gao, C., Zeng, J., Sarro, F., Lyu, M.R., King, I., 2018b. Exploring the effects of ad schemes on the performance cost of mobile phones, in: A-Mobile@ASE, ACM. pp. 13 18.

[25] Genc-Nayebi, N., Abran, A., 2017. A systematic literature review: Opinion mining studies from mobile app store user reviews. Journal of Systems and Software 125, 207 219.

[26] Gomez, L., Neamtiu, I., Azim, T., Millstein, T., 2013. Reran: Timing- and touch-sensitive record and replay for android, in: Proceedings of the 35th International Conference on Software Engineering (ICSE), IEEE. pp. 72 81.

[27] Grace, M.C., Zhou, W., Jiang, X., Sadeghi, A.R., 2012. Unsafe exposure analysis of mobile in-app advertisements, in: Proceedings of the fifth conference on Security and Privacy in Wireless and Mobile Networks (WISEC), ACM. pp. 101 112.

[28] Grano, G., Ciurumelea, A., Panichella, S., Palomba, F., Gall, H.C., 2018. Exploring the integration of user feedback in automated testing of android applications, in: IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER'18), pp. 72 83. doi:10.1109/SANER.2018.8330198.

[29] Gu, X., Kim, S., 2015. "what parts of your apps are loved by users?" (T), in: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015, pp. 760 770.

[30] Gui, J., Li, D., Wan, M., Halfond, W.G.J., 2016. Lightweight measurement and estimation of mobile ad energy consumption, in: Proceedings of the 5th International Workshop on Green and Sustainable Software, GREENS@ICSE 2016, Austin, Texas, USA, May 16, 2016, pp. 1 7.

[31] Gui, J., Mcilroy, S., Nagappan, M., Halfond, W.G., 2015. Truth in advertising: The hidden cost of mobile ads for software developers, in: Proceedings of the 37th International Conference on Software Engineering (ICSE), IEEE. pp. 100 110.

[32] Gui, J., Nagappan, M., Halfond, W.G.J., 2017. What aspects of mobile ads do users care about? an empirical study of mobile in-app ad reviews. CoRR abs/1702.07681.

[33] Guzman, E., Maalej, W., 2014. How do users like this feature? a fine grained sentiment analysis of app reviews, in: Proceedings of the 22nd International Conference on Requirements Engineering (RE), IEEE. pp. 153 162.

[34] Hsu, H., Lachenbruch, P.A., 2008. Paired t test. Wiley Encyclopedia of Clinical Trials .

[35] Iacob, C., Harrison, R., 2013. Retrieving and analyzing mobile apps feature requests from online reviews, in: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013, pp. 41 44.

[36] Iacob, C., Veerappa, V., Harrison, R., 2013. What are you complaining about?: a study of online reviews of mobile applications, in: BCS-HCI '13 Proceedings of the 27th International BCS Human Computer Interaction Conference, Brunel University, London, UK, 9-13 September 2013, p. 29.

[37] Kang, Y., Zhou, Y., Xu, H., Lyu, M.R., 2015. Persisdroid: Android performance diagnosis via anatomizing asynchronous executions. CoRR abs/1512.07950.

[38] Khalid, H., Shihab, E., Nagappan, M., Hassan, A.E., 2015. What do mobile app users complain about? IEEE Software 32, 70 77.

[39] Lin, D., Wu, X., 2009. Phrase clustering for discriminative learning, in: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL (ACL), Association for Computational Linguistics. pp. 1030 1038.

[40] Luiz, W., Viegas, F., de Alencar, R.O., Mourão, F., Salles, T., Carvalho, D., Gonçalves, M.A., da Rocha, L.C., 2018. A feature-oriented sentiment rating for mobile app reviews, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018, pp. 1909 1918.

[41] Maalej, W., Nabil, H., 2015. Bug report, feature request, or simply praise? on automatically classifying app reviews, in: 23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015, pp. 116 125.

[42] Man, Y., Gao, C., Lyu, M.R., Jiang, J., 2016. Experience report: understanding cross-platform app issues from user reviews, in: Proceedings of the 27th International Symposium on Software Reliability Engineering (ISSRE), IEEE. pp. 138 149.

[43] Martin, W., Sarro, F., Harman, M., 2016. Causal impact analysis for app releases in google play, in: Proceedings of the 24th SIGSOFT International Symposium on Foundations of Software Engineering (FSE), ACM. pp. 435 446.

[44] Martin, W., Sarro, F., Jia, Y., Zhang, Y., Harman, M., 2017. A survey of app store analysis for software engineering. IEEE Trans. Software Eng. 43, 817 847.

[45] Meng, W., Ding, R., Chung, S.P., Han, S., Lee, W., 2016. The price of free: Privacy leakage in personalized mobile in-apps ads, in: 23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016.

[46] Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 .

[47] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013b. Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems