The App Sampling Problem for App Store Mining

William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang

Department of Computer Science, University College London, London, United Kingdom

Abstract—**Many papers on App Store Mining are susceptible to the App Sampling Problem, which exists when only a subset of apps are studied, resulting in potential sampling bias. We introduce the App Sampling Problem, and study its effects on sets of user review data. We investigate the effects of sampling bias, and techniques for its amelioration in App Store Mining and Analysis, where sampling bias is often unavoidable. We mine 106,891 requests from 2,729,103 user reviews and investigate the properties of apps and reviews from 3 different partitions: the sets with fully complete review data, partially complete review data, and no review data at all. We find that app metrics such as price, rating, and download rank are significantly different between the three completeness levels. We show that correlation analysis can find trends in the data that prevail across the partitions, offering one possible approach to App Store Analysis in the presence of sampling bias.**

I. Introduction

Data availability from App Stores is prone to change: between January and September 2014 we observed two changes to ranked app availability from the Windows store: a major change, increasing the difficulty of mining information by refusing automated HTTP requests; and a minor change, extending the number of apps available. Google and Apple stores enabled mining of a large number of ranked apps during 2012 [1][2], but far fewer are currently available. At the time of mining in February 2014, we observed that the availability of app data was incomplete in all stores except for Blackberry.

This paper is concerned with reviews: user-submitted app evaluations that combine an ordinal rating and text body. Table I shows the availability of reviews at the time of writing, as well as a summarisation of the quantity of data. Availability of reviews presents a challenge to researchers working on App Store Review Mining and Analysis, because it affects generalisability. If we are only able to collect sets of the most recent data, i.e. from Google Play and Windows Phone Store, then we can only answer questions concerning this most recent data. For instance, the research question *"What is the type of feedback most talked about in app reviews?"* could not be answered using a data subset, but could be reasonably answered by collecting a random sample from all published reviews.

Recent work in App Store Review Mining and Analysis has analysed app review content and sentiment [1][3][4][5][6], extracted requirements [7][8] and devices used [9], and produced summarisation tools [2][10]. It is apparent, however, that the literature on App Store Review Mining and Analysis, with the exception of three related studies by Hoon et al. [1][11][12] and one by Fu et al. [2], uses only a subset of all reviews submitted. We therefore question whether the data provided is sufficient to draw reasonable conclusions.

TABLE I
REVIEW AVAILABILITY. NO. APPS, MOST REVIEWS FOR SINGLE APP, REVIEW AVAILABILITY AND ACCESS FOR 4 POPULAR APP STORES.

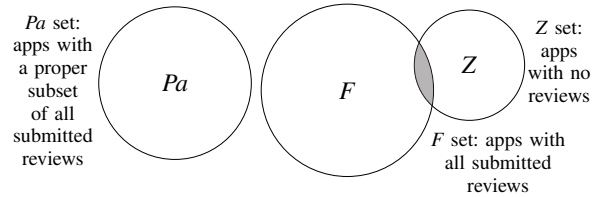| Store	Apps	Most reviews	Review availability	Method of access
Apple	1,200,000	600,000	Last release	RSS feed
Blackberry	130,000	1,200,000	Full	Web
Google	1,300,000	22,000,000	2,400	Web
Windows	300,000	44,000	36	Web

Fig. 1. **Venn diagram showing the datasets we define.**

As a way of exploring the issue empirically, we assess the level of representation an app review subset can provide. For this a full set of user reviews is needed, and so we study the Blackberry World App Store, where the full published history is available.

The contributions of this paper are as follows:
1) We survey the literature in the field of App Store Review Mining and Analysis, and identify the prevalent problem of partial datasets.
2) We present empirical results that highlight the pitfalls of partial datasets and consequent sampling bias.
3) We mine and analyse a dataset of apps and user requests, using manually validated automatic extraction. We illustrate the way correlation analysis may potentially ameliorate sampling issues in App Store Review Mining and Analysis.

II. The App Sampling Problem

We define the following sets, shown in Fig. 1, in order to describe different levels of data completeness.
(*Pa*) **Partial**: The set of apps and their reviews for which the dataset contains only a proper subset of the reviews submitted to the App Store.
(*F*) **Full**: The set of apps and their reviews for which the dataset contains all of the reviews submitted to the App Store.
(*Z*) **Zero**: The set of apps which have no reviews. Strictly speaking Z may intersect F, as some apps have never been reviewed by a user.
(*A*) **All**: The combination of the above datasets; includes all mined data ($A = Pa \cup F \cup Z$).

Authors [Ref], Year	Store	No. apps	No. reviews	Type
Hoon et al. [11] 2012	A	17,330	8,701,198	F
Vasa et al. [12] 2012	A	17,330	8,701,198	F
Hoon et al. [1] 2013	A	17,330	8,701,198	F
Iacob et al. [7] 2013	G	270	137,000	Pa
Galvis Carreño et al. [8] '13	G	3	327	Pa
Khalid [3] 2013	A	20	6,390	Pa
Fu et al. [2] 2013	G	171,493	13,286,706	F
Pagano et al. [4] 2013	A	1,100	1,100,000	Pa
Iacob et al. [5] 2013	G	161	3,279	Pa
Khalid [13] 2014	A	20	6,390	Pa
Chen et al. [10] 2014	G	4	241,656	Pa
Guzman et al. [6] 2014	A,G	7	32,210	Pa
Khalid et al. [9] 2014	G	99	144,689	Pa
This paper 2015	B	15,095	2,729,103	A

There has been a surge of work recently in the field of App Store Mining and Analysis, much of which is focussed on app reviews. 9 out of 13 past studies we found on App Store Review Mining and Analysis use *Pa* sets of data, drawn from the most recent reviews of the most popular apps at the time of sampling. A summary of the work to date on App Review Mining and Analysis is presented in Table II, and a further discussion of related App Store research is presented in Section VII.

Following from the data accessibility summarised in Table I in the four App Stores, sets are available under the following conditions at the time of writing:

Apple App Store: *F* set available for apps with one release, or where only the most recent release has been reviewed; *Pa* set available for apps with multiple reviewed releases.

Blackberry World App Store: *F* set available for all apps.

Google Play: Data available for a maximum of 480 reviews under each star rating via web page (2,400 total) per app; hence *F* set available for apps with fewer than 481 reviews under each star rating, *Pa* set available otherwise.

Windows Phone Store: Data available for a maximum of 36 reviews per app; hence *F* set available for apps with less than 37 reviews, *Pa* set available otherwise.

Historically, greater access appears to have been available. Further data access is also available through 3rd party data collections, subject to their individual pricing plans, terms and conditions; such collections are out of the scope of this paper as they are not freely available, and have not been used in the literature.

Availability of App Store data is likely to become a pressing problem for research if we cannot find ways to mitigate the effects of "enforced" sampling bias: sampling in which sample size and content are put outside of experimental control. This may be for reasons unavoidable (App Store data availability), pragmatic (memory or framework limitations on loading large dynamic web pages), or otherwise. We call this the *App Sampling Problem*.
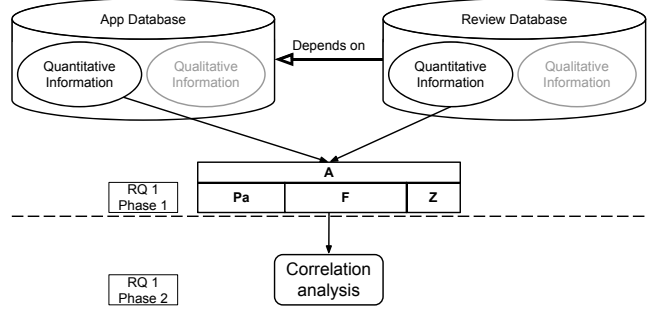

Fig. 2. **Process for comparing data subsets.** The boxes indicate the phases of RQ1 that each part of the diagram corresponds with.

III. METHODOLOGY

In this section we provide details of the metrics used, research questions, algorithms used for collecting the dataset, steps for preprocessing the text, and topic modelling application details.

A. Metrics

The goal of this paper is to compare the datasets defined in Section II, and for this comparison the following app properties are used:

(P)rice: The amount a user pays for the app in USD in order to download it. This value does not take into account in-app-purchases and subscription fees, thus it is the 'up front' price of the app.

(R)ating: The average of user ratings made of the app since its release.

(D)ownload rank: Indicates the app's popularity. The specifics of the calculation of download rank vary between App Stores and are not released to the public, but we might reasonably assume the rank approximates popularity, at least on an ordinal scale. The rank is in descending order, that is it increases as the popularity of the app decreases: from a developer's perspective, the lower the download rank the better.

(L)ength of description: The length in characters of the description, after first processing the text as detailed in Section III-E, and further removing whitespace in order to count text characters only.

(N)umber of ratings: The total number of ratings that the app has received. A rating is the numerical component of a review, where a user selects i.e. a number from 1 to 5.

B. Research Questions

RQ1: How are trends in apps affected by varying dataset completeness?

To answer this question we follow the illustration in Fig. 2 in which Phase 1 in the figure corresponds with RQ1.1 and Phase 2 in the figure corresponds with RQ1.2.

RQ1.1: Do the subsets Pa, F and Z differ? The mined dataset will first be separated into the subsets of *Pa*, *F* and *Z*, as illustrated in Fig. 2.

We will then compare the datasets using the metrics (P)rice, (R)ating, (D)ownload rank, (L)ength of description and (N)umber of ratings with a 2-tailed, unpaired Wilcoxon test [14]. The Wilcoxon test is appropriate because the metrics we are investigating are ordinal data, and we therefore need a non-parametric statistical test that makes few assumptions about the underlying data distribution. We make no assumption about which dataset median will be higher and so the test is 2-tailed; the test is unpaired because the datasets contain entirely different sets of apps. We test against the Null Hypothesis that the datasets in question are sampled from distributions of the same median, and a p-value of less than 0.05 indicates that there is sufficient evidence to reject this null hypothesis with a maximum of 5% chance of a Type 1 error. Because multiple tests are performed, we apply Benjamini-Hochberg correction [15] to ensure we retain only a 5% chance of a Type 1 error.

We investigate the effect size difference of the datasets using the Vargha-Delaney $\hat{A}_{12}$ metric [16]. Like the Wilcoxon test, the Vargha-Delaney $\hat{A}_{12}$ test makes few assumptions and is suited to ordinal data such as ours. It is also highly intuitive: for a given app metric (P, R, D, L or N), $\hat{A}_{12}$(A, B) is an estimate of the probability that the value of a randomly chosen app metric from Group A will be higher than that of a randomly chosen app metric from Group B.

RQ1.2: Are there trends within subsets Pa, F and Z? We will compare the trends within datasets *Pa*, *F*, *Z* and *A* using three correlation analysis methods: Spearman's rank correlation coefficient, Pearson's correlation coefficient and Kendall's Tau rank correlation coefficient. These methods will be run on each pair of the metrics P, R, D, L and N to check for correlations in subsets of the data, which we can compare with the overall trends from the *A* set.

RQ2: What proportion of reviews in each dataset are requests?

Once a set of app reviews has been obtained, some applications of App Store Analysis involve focussing on a well-defined subset of these reviews. For example, this is useful for development prioritisation, bug finding, inspiration and otherwise requirements engineering ([2][3][5][6][7][8][10]). We explore the subset of reviews that contain user requests, as this is the most studied in the literature.

RQ2.1: What proportion of user reviews does the Iacob & Harrison [7] algorithm identify? We use the algorithm proposed by Iacob and Harrison [7] to identify requests in the set of mined reviews. Reviews are treated to the text pre-processing algorithm detailed in Section III-E. The algorithm detects requests on a per-sentence basis by matching key words in a set of linguistic rules; in this case the sentences are the entire review bodies as we remove punctuation as part of the text pre-processing. This does not affect the matching except to reduce the number of possible matches per review to 1.

RQ2.2: What is the Precision and Recall of the extraction algorithm? We assess the Precision and Recall of the request extraction algorithm, as well as random guessing.

We sample 1000 random reviews from each of 4 sets: *Pa* (all reviews; requests), *F* (all reviews; requests). The sample of 1000 is representative at the 99% confidence level with a 4% confidence interval [17]. Under the criterion that "*the user asks for ('requests') some action to be taken*", each sampled review is manually assessed as either a valid request or an invalid request.

RQ3: How are trends in requests affected by varying dataset completeness?

RQ1 seeks to establish an experimental approach to handling partial data, which we now apply to an instance, specifically request analysis. To answer RQ3 we will compare user requests from data subsets of varying completeness. We will then compare the trends in each group of requests using correlation analysis.

We use the requests and try to learn about their content. The set of requests contains raw text data, and so to compare them we first train a topic model on the set of app descriptions and identified requests, as detailed in Section III-F. It is advantageous to use a topic modelling approach as the output of such algorithms is a probabilistic distribution of latent 'topics' for each input 'document'. Such output allows for intersections of app descriptions and user requests that share content to be computed through use of a threshold probability value. Topic modelling is used throughout the literature in app analysis for categorisation ([18]), summarisation ([2][10]), data extraction ([6][8]), and API threat detection ([19]).

A topic's 'strong contribution' to an app description or user request indicates that the topic falls above the topic likelihood threshold for that particular document. Experiments in this section use a topic likelihood threshold of $t = 0.05$, as is the standard in the literature. For comparison, we define 3 metrics at the topic level:

Ta: App prevalence: the number of app descriptions to which the topic strongly contributes.
Tr: Request prevalence: the number of requests to which the topic strongly contributes.
Td: Download rank: the median download rank of the apps to which the topic strongly contributes.

Results in this section use a topic likelihood threshold of $t = 0.02$ to ensure a large enough set of topics for the statistical set comparison tests Wilcoxon and $\hat{A}_{12}$. Using the set of topics from RQ2.2, we compare requests between *Pa*, *F* and *A* datasets using both the topics themselves and the three topic metrics. Because RQ3 concerns request properties and trends, we exclude the *Z* set for this question, since it cannot contain requests.

RQ3.1: Do the request sets from Pa, F and A differ? To answer this question we run the same 2-tailed, unpaired Wilcoxon test and $\hat{A}_{12}$ metric used in RQ1.1. We apply the statistical tests to the Ta, Tr and Td distributions between *Pa*, *F* and *A* datasets. This shows us the difference between the dataset distributions, and indicates how the effect size differs. We apply Benjamini-Hochberg [15] correction for multiple tests to ensure we retain only a 5% chance of a Type 1 error.

RQ3.2: Are there trends within the request sets of Pa, F and A? We compare the trends of requests within each set by applying correlation methods to each pair of the topic metrics app prevalence, request prevalence and download rank, which are formally defined in Section III-F. We run the three correlation methods (Spearman, Pearson and Kendall) as used in RQ1.2, on the topic metrics Ta, Tr and Td. This allows us to identify how requests are affected by varying dataset completeness, and to establish whether we can learn anything from incomplete datasets.

C. Mining Process

We use a process for mining apps that is similar to the 4-step approach used by Harman et al. [20], as shown in Fig. 3. The first phase extracts the list data for the most popular apps in the store, providing locations from which to download app and review data for each individual app. In the second phase we download the raw html data for apps and reviews for each app, and proceed to use data mining in the third phase to extract quantitative data, such as price and rating; and qualitative data such as the name and description (or review) body. In the fourth phase we record the extracted information in databases to enable further analysis. The rest of this section explains the first three steps of our extraction process in more detail.

Data Mining Phase 1 (Extraction of App List): We use the `spynner`[1] framework to continually load more apps to the 'most popular' list, but available memory becomes a limiting factor as the page size grows rapidly. On a machine with 4GB of RAM, we are able to capture a list of between 20k and 24k apps before the framework runs of out memory. We therefore set a limit on the number of captured apps of 20k. Since 2011 the number of apps in the store has grown from around 60k to 230k [17], and we are therefore unable to capture the full list.

Data Mining Phase 2 (Raw App and Review Data Download): We visit each app in the list extracted by Phase 1, and download the html page which holds the app metadata. For review data, we use the `spynner` framework to continually load more reviews from the page. This has the same memory limitation as Phase 1, allowing us to capture between 4k and 4.5k reviews per app before the framework runs out of memory; we therefore limit the number of captured reviews to 4k per app, and it is this limitation that causes us to have a *Pa* dataset; the set of apps for which we have mined only a subset of the available reviews.

Spam is common in app reviews, and can occur for reasons of competition i.e. to boost one's own app, or to negate a competitor's app. Some App Stores have methods in place to help prevent spam, but these methods are not publicly available, and they can never be 100% effective. Jindal and Liu [21] found that online reviews that are identical to another review are almost certainly spam, and are also the largest portion of spam by type. We therefore filter the reviews for duplicates, including only one copy of each unique review.

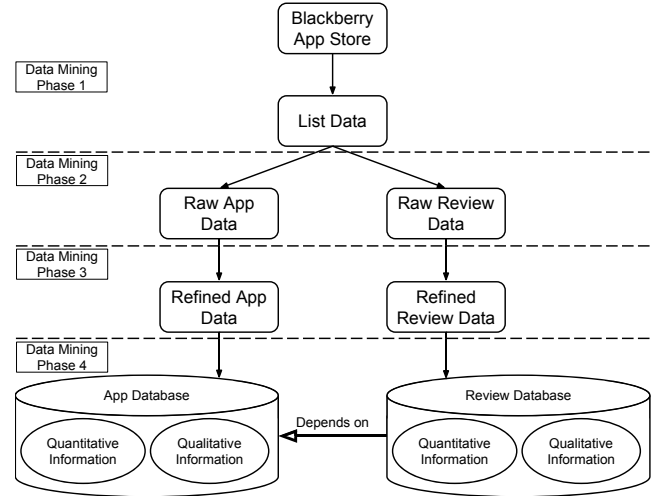[1] https://pypi.python.org/pypi/spynner

Fig. 3. **App and Review mining process** showing the 4-phase approach to data collection used as part of this study. The boxes indicate data mining phases which are described in Section III-C.

A unique review is defined by the rating, review body and author name. We do not use more sophisticated spam detection and filtering methods in this paper because there is no prescribed standard, and spam detection is not the aim of our study.

Data Mining Phase 3 (Parsing): We parse the raw data for a set of unique searchable signatures, each of which specifies an attribute of interest. These patterns enable us to capture app information such as price, rating, category, description, while other information such as the rank of downloads and the identifier were established in Phase 1. The unique searchable signature for rating is as follows: `awwsProductDetailsContentItemRating`. The mined review data is parsed in the same way, to capture each review's rating, body and author attributes; although the author information is only used to ensure that no duplicates are recorded as a means of reducing spam.

Data Mining Phase 4 (Database): We store the parsed app data in an app database, and the parsed review data in a review database in which each entry refers to an app identifier.

The approach is applicable to any App Store, with sufficient changes to the sections parsing the web pages. However in other App Stores the initial list popularity information from Phase 1 would need to be adapted, as other stores do not provide a global popularity ranking, but instead separate free and paid, or by category.

D. Dataset

The dataset used in this study was mined during February 2014, and is presented in Table III. Using the process detailed in Section III-C, a snapshot of the top 20,256 most popular apps was taken, and the corresponding web pages were subsequently parsed for descriptions and metadata such as price and average rating.

TABLE III
DATASETS USED THROUGHOUT STUDY, MINED FROM BLACKBERRY
WORLD APP STORE AND PARTITIONED INTO THE SETS Pa, F, Z AND A.

Dataset	Apps	Reviews
Pa	5,422	1,034,151
F	6,919	1,694,952
Z	2,754	0
A	15,095	2,729,103

TABLE IV
TOPIC MODELLING SETTINGS APPLIED THROUGHOUT THE STUDY.

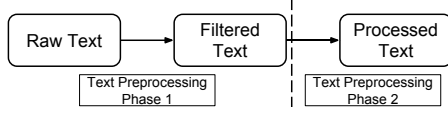| Setting	Value
Number of latent topics K	100
Prior on per-document topic distribution α	50
Prior on per-topic word distribution β	0.01
Number of Gibbs sampling iterations	1000

Fig. 4. **Text pre-processing process.** Shows the 2-phase approach to text pre-processing used throughout this study. The boxes indicate text pre-processing phases which are detailed in Section III-E.
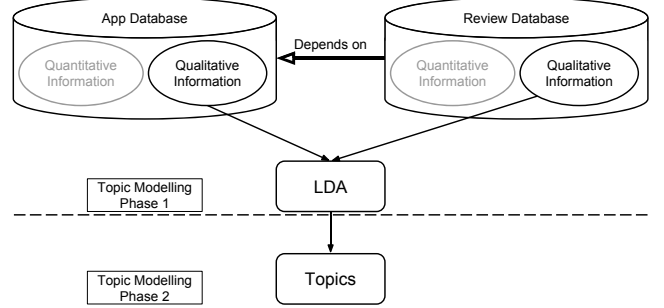

Fig. 5. **Topic modelling process**. Illustration of the topic modelling process used in this study. The boxes indicate topic modelling phases which are detailed in III-F.

Errors in app pages such as negative ratings, negative number of ratings and empty app ids, led to the exclusion of a number of apps. We judged that the anomalies would affect the results of our experiments. The final number of apps used in the study is therefore 15,095[2].

This full set of apps is the A set. As detailed in Table III, it is split into Pa, F and Z subsets according to the definitions in Section II.

E. Text Preprocessing

Through this study, the mined qualitative data including app description text and review body text, is treated to the pre-processing steps shown in Fig. 4.

Text Preprocessing Phase 1 (Filtering): Text is filtered for punctuation and the list of words in the English language `stopwords` set in the Python NLTK data package[3], and is cast to lower case.

Text Preprocessing Phase 2 (Lemmatisation): Each word is transformed into its 'lemma form' using the Python NLTK `WordNetLemmatizer`. This process homogenises singular/plural, gerund endings and other non-germane grammatical details, yet words retain a readable form.

F. Topic Modelling

This section describes the topic modelling process used in the study, and defines terms used later on. The main process is illustrated in Fig. 5, and will be further described below.

Topic Modelling Phase 1: The text from each app description and user review that has been identified as a request is processed as described in Section III-E. This text is fed into a modified version of Latent Dirichlet Allocation (LDA) [22], which enhances performance through parallelisation [23]. This is applied using Mallet [24], a Java tool for applying machine learning and language processing algorithms.

[2]Data from this paper is available at http://www0.cs.ucl.ac.uk/staff/W. Martin/projects/app_sampling_problem/.

[3]`nltk.corpus.stopwords.words('english')`

We chose settings commonly used in the literature as the goal of this study is not to find optimal topic modelling settings for such a study, but to provide the means to easily replicate the study. With the exception of the threshold on per-document likelihood, which is chosen on a per-experiment basis, the fixed settings specified in Table IV were used. We experimented with settings of $K = 100$, 200, 500 and 1000 topics, and found that the settings of $K = 500$ and $K = 1000$ led to over-specialised topics which reflected the descriptions of individual apps. The settings of $K = 100$ and $K = 200$ enabled more generalisation throughout the topics, and we selected $K = 100$ as we judged that it was the lowest setting without any app-specific topics.

Topic Modelling Phase 2: The output of LDA includes a list of topic likelihoods for each document (in this case an app description or user request). From this we compute a list for each topic, of the app descriptions and user requests for which the topic-document likelihood exceeds a threshold.

Topic modelling with LDA leads to topics that reflect the source corpus, and so inevitably if spam exists in any of the user requests or app descriptions, it will also appear in some topics. For this reason we manually verified the 100 trained topics and classified them as valid or invalid, with the criteria that "the top 5 terms should refer to some functional or non-functional app property or properties". Classification results are presented in Table V. An example valid topic includes the top terms: {gps, speed, location, track, distance...}, and an example invalid topic includes the topic includes the top: terms {great, much, awesome, worth, well...}. Henceforth only the 80 valid topics are used for comparison.

MANUAL TOPIC CLASSIFICATION RESULTS. ONLY TOPICS THAT ARE
CLASSIFIED AS VALID ARE USED IN THIS STUDY.

Class	Description	Count
Valid	Defines one or more functional or non-functional properties of an app.	80
Review	Words highly coupled to review text such as [error, load, sometimes].	8
App	Words highly coupled to app description text such as [please, review, feedback].	6
Spam	Highlights spam in app descriptions and reviews used for advertising.	1
Other	Does not fit into another category but is not valid.	5
Total	All topics	100

TABLE VI
RQ1.1: COMPARISON RESULTS BETWEEN APP DATASETS. WE
COMPARE (P)RICE, (R)ATING, (D)OWNLOAD RANK, (L)ENGTH OF
DESCRIPTION AND (N)UMBER OF RATINGS IN SETS *Pa*, *F* AND *Z* USING
VARGHA AND DELANEY'S $\hat{A}_{12}$ EFFECT SIZE COMPARISON. LARGE
EFFECT SIZES FROM THE $\hat{A}_{12}$ TEST (≥ 0.8) ARE MARKED IN BOLD.

Datasets	P	R	D	L	N
Pa - F	0.533	0.529	0.564	0.554	0.630
Pa - Z	0.514	**0.965**	**0.834**	0.718	**1.000**
F - Z	0.547	**0.966**	**0.811**	0.763	**1.000**

IV. RESULTS

RQ1: How are trends in apps affected by varying dataset completeness?

RQ1.1: Do the subsets Pa, F and Z differ?

The Wilcoxon test results for metrics P, R, D, N and N
between the sets *Pa*, *F* and *Z* were all < 0.001, and remained
below 0.05 after we applied Benjamini-Hochberg correction
for multiple tests. This shows that there is a significant
difference between sets *Pa*, *F* and *Z*. The $\hat{A}_{12}$ effect size
comparison results in Table VI show that the *Pa* and *F* sets
have a small effect size difference: hence, *Pa* and *F* are
unlikely to yield different results for the metrics tested.

This result is a contrast to the comparisons between sets
Pa and *Z*, and *F* and *Z*. The *Z* set contains only apps that
have not been reviewed and therefore have no rating, and
hence have a large effect size difference with the N (number
of ratings) metric. However, there is also a large effect size
difference between the rating, download rank and description
length between *Z* and the other sets. This shows that the *Z*
set is very different from the other sets and indicates that it
should be treated separately when analysing App Store data.

All sets have a small effect size difference when comparing
P (price), which might be expected because 90% of the apps
in the set *A* are free.

RQ1.2: Are there trends within subsets Pa, F and Z?

The results in Table VII serve to further distinguish the
need to separate the *Z* set when performing app studies, as it
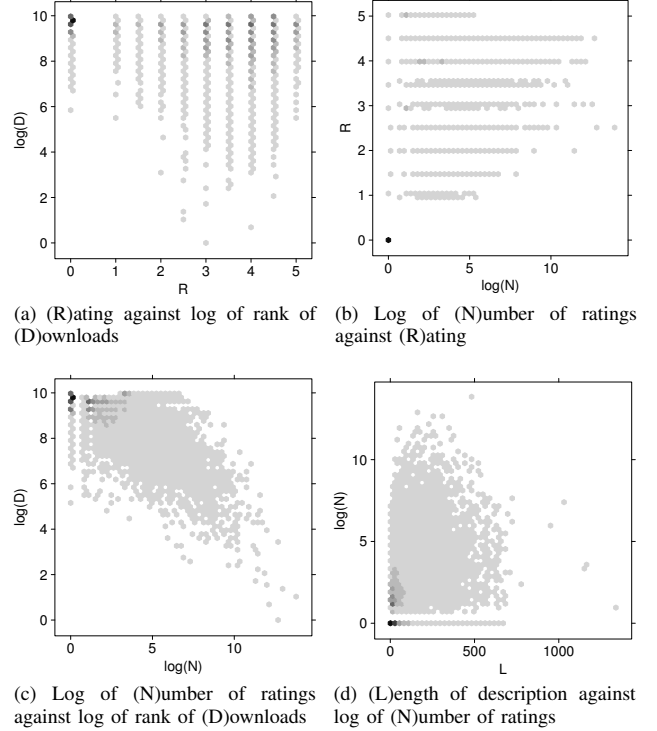possesses almost none of the trends of the other sets.

(a) (R)ating against log of rank of (D)ownloads

(b) Log of (N)umber of ratings against (R)ating

(c) Log of (N)umber of ratings against log of rank of (D)ownloads

(d) (L)ength of description against log of (N)umber of ratings

Fig. 6. **RQ1.2: Scatter plots for for (A)ll dataset** for strong and significant results. Points are clustered in cells whose darkness represents their density.

Indeed, almost all correlation results from the *Z* set were
not statistically significant (marked with '-' in the table).
Conversely, most results from the other sets were statistically
significant, and the following (marked in bold), were strong:

*a) Inverse **RD** correlation:* This indicates that as the
(**R**)ating of apps increases, the (**D**)ownload rank decreases.
This result was not particularly strong, but we notice that it
was present in the *A* and *Z* sets only. This result is caused by
large numbers of non-rated apps in the *Z* dataset.

*b) Positive **NR** correlation:* This indicates that as the
(**N**)umber of ratings increases, the (**R**)ating of apps increases.
The trend was present in the *A* set only, and non-existent in the
Pa, *F* or *Z* sets. Similar to the RD correlation result, this result
is caused by the combination of non-rated and rated apps.

*c) Inverse **ND** correlation:* This indicates that as the
(**N**)umber of ratings increases, the (**D**)ownload rank decreases,
which means that the app is more popular.

*d) Positive **LN** correlation:* This indicates that as the
(**L**)ength of descriptions increases, the (**N**)umber of ratings
for the app increases. This result was consistent across both
Pa and *F* datasets, and stronger overall, perhaps making an
argument to also look for trends across all mined data.

We analyse these correlations in more detail by examining Fig. 6, which shows that the RD and NR correlations exist
only because of a large number of non-rated, and therefore
zero-rated, apps from the *Z* dataset. Results such as these
emphasise the need to consider *Pa*, *F* and *Z* datasets separately.

TABLE VII
RQ1.2: Correlation results in each of the 4 datasets between (P)rice, (R)ating, (D)ownload rank, (N)umber of ratings and (L)ength of description. Datasets are defined in terms of review set completeness: (*Pa*)rtial, (*F*)ull, (*Z*)ero and (*A*)ll. Results are presented for Spearman's, Pearson's and Kendall's Tau correlation coefficients, in each case only where the p-value is less than 0.05. Results in bold have strong correlation coefficients.

Dataset	Correlation	PR	PD	RD	NP	NR	ND	LP	LR	LD	LN
	Spearman	0.163	0.120	-0.062	0.107	0.035	**-0.641**	0.234	0.094	-0.213	0.391
Pa	Pearson	0.097	0.083	-0.077	-	-	-0.116	0.199	0.068	-0.149	0.082
	Kendall	0.140	0.096	-0.044	0.089	0.031	-0.471	0.190	0.069	-0.141	0.272
	Spearman	0.253	0.385	-0.029	0.117	0.057	-0.352	0.254	0.147	-0.056	0.317
F	Pearson	0.104	0.231	-0.035	-	0.082	-0.234	0.162	0.131	-	0.095
	Kendall	0.216	0.309	-0.023	0.094	0.047	-0.239	0.204	0.107	-0.038	0.215
	Spearman	-0.063	-	-0.228	-	-	-	0.260	-0.200	-	-
Z	Pearson	-	-	-0.220	-	-	-	0.277	-0.116	-	-
	Kendall	-0.062	-	-0.184	-	-	-	0.214	-0.163	-	-
	Spearman	0.200	0.220	-0.307	0.141	0.452	**-0.568**	0.268	0.272	-0.223	0.456
A	Pearson	0.086	0.129	-0.391	-	0.025	-0.078	0.185	0.223	-0.158	0.057
	Kendall	0.17	0.177	-0.224	0.116	0.336	-0.439	0.217	0.197	-0.152	0.320

TABLE VIII
RQ2.1: Request datasets extracted from *Pa*, *F* and *A* review sets using the Iacob & Harrison [7] algorithm. Requests indicates the number of reviews matching a request rule, and proportion indicates the number of requests relative to the review set they were extracted from.

Dataset	Apps	Reviews	Requests	Proportion
Pa	5,422	1,034,151	32,453	3.14%
F	6,919	1,694,952	74,438	4.39%
Z	2,754	0	0	0%
A	15,095	2,729,103	106,891	3.92%

> **RQ1: How are trends in apps affected by varying dataset completeness?** *There is a significant difference between Pa, F and Z datasets, in some cases with a large effect size; the trends observed differ between datasets, especially when Z is included.*

RQ2: What proportion of reviews in each dataset type are requests?

RQ2.1: What proportion of user reviews does the Iacob & Harrison [7] algorithm identify?

Details of the extracted requests can be found in Table VIII. The proportion of extracted requests is lower than that reported by the previous study of Iacob & Harrison [7], which can be attributed to two major differences in this study: **1) Blackberry World App Store is used in this study instead of Google Play.** It may be that user behaviour in reviews is different, manifesting in less requests. It may also be that users use different words or phrases when making requests, and so the request extraction algorithm's recall is diminished. **2) Our dataset contains 19 times more reviews and 56 times more apps.** It therefore includes many more lower ranked and less popular apps; it is possible that users make fewer requests in their reviews of such apps.

RQ2.2: What is the Precision and Recall of the extraction algorithm?
The assessment was completed by one author in two days. Results from the assessment can be found in Table IX, and the computed Precision, Recall and F-Measure are found in Table X. The resulting Precision is low, yet Recall is very high; the Iacob & Harrison algorithm [7] performs over 11 times better than random guessing.

An example TP (true positive) request and FP (false positive) request can be read below.

TP request: "Would be nice to be able to access your account and rate the movies. Please include for next update."

FP request: "Amazing app every thing u need to know about real madrid is in this app and it deserve more than five stars"

A general observation from the sampling process was that the 'identified as requests' reviews were not short, and always contained 'request-like' words. However, there was a large proportion of (FP) reviews which contained words like 'need', but did not ask for any action to be taken by the developers. Handling such cases is a challenge for content analysis of user reviews such as Iacob & Harrison algorithm [7], and may require more sophisticated analysis, and perhaps the inclusion of sentiment, to properly deal with.

The set of FN (false negative) reviews, which were not identified as requests but were manually assessed to contain a request, asked for work to be done in different ways. Linguistic rules could be added to encompass the FN cases, but this would run the risk of further lowering the Precision. An example FN request can be read below.

FN request: "Sharp, but layout can be worked on!"

The high Recall result shows that we are able to reduce the set of all reviews to a much smaller set that is more likely to contain requests, and this process excludes very few requests falsely. We suggest that without a proven better performing method for data reduction of user reviews, it is sufficient to use the set of all requests identified by the Iacob & Harrison algorithm [7] for analysis.

TABLE IX
RQ2.2: Manual assessment of reviews. A sample of 1000 random reviews was taken from each of 4 sets, and each review was manually classified as containing a request for action to be taken or not. The sets used were the reviews and requests from each of the (Pa)rtially complete dataset and the (F)ully complete dataset.

Dataset	Type	Population	Sample	TP	FP	Precision
Pa	Request	32,453	1000	279	721	0.279
Pa	Review	1,034,151	1000	14	986	0.014
F	Request	74,438	1000	347	653	0.347
F	Review	1,694,952	1000	29	971	0.029

TABLE X
RQ2.2: Assessment of request algorithm. Using the results from the manual assessment in Table IX, we compute the Precision, Recall and F-Measure for the (Pa)rtially complete dataset and the (F)ully complete dataset.

Dataset	TP	FP	FN	Precision	Recall	F-Measure
Pa	279	721	10	0.279	0.965	0.433
F	347	653	9	0.347	0.975	0.512

TABLE XI
RQ3.1: Comparison results between request sets. We compare topic app prevalence, request prevalence and median download rank between sets Pa, F and A using Vargha and Delaney's $\hat{A}_{12}$ effect size comparison. Large effect sizes from the $\hat{A}_{12}$ test (≥ 0.8) are marked in bold.

Datasets	Ta	Tr	Td
Pa - F	0.794	**0.974**	**0.999**
A - F	**0.831**	**0.839**	**0.874**
A - Pa	**0.914**	**0.998**	**0.971**

> **RQ2: What proportion of reviews in each dataset type are requests?** Less than 3.92% of A, 3.14% of Pa and 4.39% of F reviews are requests in the Blackberry dataset, as we have found the algorithm to have a Recall > 0.96 and Precision < 0.350.

RQ3: How are trends in requests affected by varying dataset completeness?

RQ3.1: Do the request sets from Pa, F and A differ? The Wilcoxon test results for metrics Ta (topic app prevalence), Tr (topic request prevalence) and Td (topic median download rank) between the sets Pa, F and A show that there is a significant difference between sets Pa, F and A. The $\hat{A}_{12}$ test results in Table XI also show that there is a large effect size difference for Ta, Tr and Td between the three sets.

We have seen from the results for RQ1.1 in Table VI that there is a significant distribution difference, but a small effect size difference between the description lengths in Pa and F. This means that Pa and F are unlikely to yield different description lengths. It can seem surprising, therefore, that there is a large effect size difference between the app prevalence for topics between the Pa and F sets.

Recall that app prevalence for a topic means the number of app descriptions to which the topic strongly contributes. However, this result is explained by the size difference of the sets: it stands to reason that in a set with more apps, a topic would be featured in more app descriptions than in the smaller set. This effect can be seen in greater magnitude with the larger request prevalence effect size difference. The Pa set has a large effect size difference in Tr from both the A set and the F set. These results again show that topics are used more frequently in the larger of the sets, an expected result.

What is more surprising are the results for Td. Table XI shows that the distributions of topic download rank in the three sets are very different, and have a large effect size difference. Bearing in mind that the A set is the combination of apps from Pa, F and Z, the A - Pa result suggests that the Pa set's median download rank is different from the overall median, while the F set's median is closer. However, the Pa - F results are surprising as they mean that for the same topics, one set leads to apps with greater median download ranks than the other; yet the results for RQ1.1 in Table VI showed a small effect size difference in download rank for Pa - F.

One possible explanation is that the topic download ranks are exaggerating the difference in distribution medians. Because of the way the topic model is trained, each app description must have at least one assigned topic; likely more than one as the Dirichlet prior α is set to 50 [22]. Therefore, the entire set of app descriptions must be represented by only the 100 topics (80 of which are used here as explained in Section III-F). It is not hard to imagine that the median download rank of apps in these 80 topics could be very different for Pa, F and A, even if the distributions of apps have small effect size differences in D.

RQ3.2: Are there trends within the request sets of Pa, F and A? We can see from Fig. 7 that the F set has a higher median app prevalence and request prevalence for topics. That is, topics contribute in a greater number of apps and requests on average in the F set than the Pa set.

The results presented in Table XII show that there is a strong negative correlation between the prevalence of topics in apps and requests. This is evidence that users tend to request things to a greater extent when they are present in a smaller proportion of apps; a linear relationship is suggested, because the correlation coefficient is strong with Pearson's method. We can see from the graphs in Fig. 7 that the scatter plot of Ta-Tr from the F set resembles the Pa set, but appears more stretched out and higher due to the greater number of apps and requests in the F set.

One might expect there to be a strong correlation between popularity and request prevalence, indicating that users request things present in more popular apps, but this is not the case, as there was no significant correlation between the two metrics. An interpretation of this result is that while users do not discriminate in terms of popularity, they do value diversity, hence the results of the negative prevalence correlation.

Dataset	Correlation	Ta - Tr	Ta - Td	Tr - Td
Pa	Spearman	-0.341	-0.230	-
	Pearson	**-0.642**	-0.253	-
	Kendall	-0.243	-0.166	-
F	Spearman	**-0.596**	-	-
	Pearson	**-0.719**	-	-
	Kendall	-0.446	-	-
A	Spearman	**-0.554**	-	-
	Pearson	**-0.749**	-	-
	Kendall	-0.407	-	-

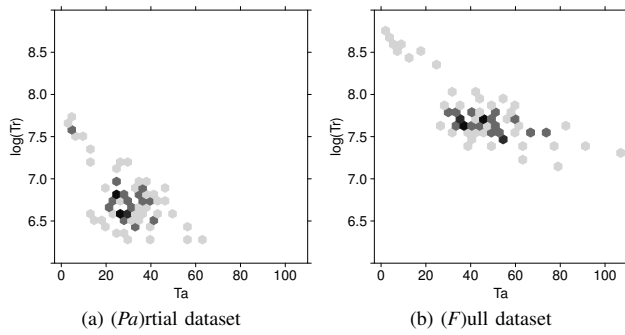(a) (*Pa*)rtial dataset   (b) (*F*)ull dataset

Fig. 7. **RQ3.2: Scatter plots for App Prevalence (Ta) against Request Prevalence (Tr)** for *Pa* and *F* datasets. Points are clustered in cells whose darkness indicates their density.

An important finding from these results is that the correlation between app and request prevalence exists not only in *Pa* and *F* datasets, but is actually stronger overall in the *A* dataset. This demonstrates that although the *Pa* dataset is incomplete and has different properties, the trends are consistent showing that we are still able to learn things from the data.

> *RQ3: How are trends in requests affected by varying dataset completeness? Trends in requests appear more robust to app sample bias than trends in apps; we have found a strong inverse linear correlation between topic prevalence in apps and requests, that is consistent in Pa, F and A datasets.*

V. Actionable Findings for Future App Store Analysis

In this paper we have presented empirical evidence that indicates that the partial nature of data available on App Stores can pose an important threat to the validity of findings of App Store Analysis. We show that inferential statistical tests yield different results when conducted on samples from partial datasets compared to samples from full data sets; even if we were to exhaustively study the entire partial data set available, we will be studying a potentially biased sample.

The findings reported in this paper suggest that this will be a potent threat to validity. Naturally, where researchers have full data sets available they should sample (randomly) from these or base their results on the exhaustive study of the entire dataset. However, this raises the uncomfortable question: "What should we do when only a partial dataset is available?". This question is uncomfortable, because partial datasets are so prevalent in App Store Analysis, as the survey in this paper indicates.

Clearly, researchers need to augment their research findings with an argument to convince the reader that any enforced sampling bias is unlikely to affect their research conclusions and findings. This may be possible in some cases, since the sampling bias is often at least known, and sometimes quantifiable. For example there may be simply a bias on either recency or on popularity. Alternatively, researchers may choose to constrain study scope, making claims only about populations for which the sample is unbiased (e.g., all recent or popular apps on the store(s) studied).

These are rather generic and standard approaches to handling sampling bias. Our analysis of partial and complete data from the Blackberry World App Store indicates a further possibility. Tentatively, our results suggest that correlation analysis may prove to be less sensitive to biasing effects of enforced sampling, compared to inferential statistical testing for significant differences between biased samples.

Ideally, one would like to see further studies of other App Stores to confirm these results. Unfortunately, many App Stores do not make full information available, as the Blackberry World App Store did at the time we studied it. Nevertheless, where full information is available, further replication studies are desirable: correlation analysis of App Stores provides a potential tool to understand the relationship between the technical software engineering properties of apps, and the users' perceptions of these apps. Understanding such relationships is one of the motivations for the excitement and rapid uptake of App Store Analysis.

VI. Threats to Validity

Our paper is primarily concerned with addressing threats to validity, yet it also may have some of its own. In this section we discuss the threats to the validity of our experiments based on *construct*, *conclusion* and *external* validity.

Our construct validity could be affected by issues with the data we used in our experiments, which was gathered from the Blackberry World App Store. We therefore rely on the maintainers of the store for the reliability and availability of raw data. Due to the large-scale and automated nature of data collection, there may be some degree of inaccuracies and imprecisions in the data. However, we have focussed our observations across large sets of data, rather than detailed observations from individual apps, which provides some robustness to the presence of inaccuracies in the data. We also make the assumption when processing the (N)umber of ratings for an app, that reviews are not removed from the store. This would cause an app that is reviewed to appear non-rated.

Were an app with many reviews to have some removed, it is unlikely that this would impact the overall findings, as the scale of the number of ratings provides some robustness to small changes. To the best of our knowledge, however, reviews are neither removed nor changed.

Our conclusion validity could be affected by the human assessment of topics and requests in answer to RQ2. To mitigate this threat, we plan to conduct replications using a larger number of participants.

With regard to external threats, we return once again to the dataset. We mined a large collection of app and review data from the Blackberry World App Store, but we cannot claim that our results generalise to other stores such as those owned by Google or Apple. Rather, our study discusses the issues that can arise from using biased subsets of reviews, such as the kind available from App Stores; and we use the Blackberry dataset to demonstrate our approach at mitigating the bias, in order that we can learn from the data.

VII. RELATED WORK

App Store Repository Mining is a recently introduced field that treats data mining and analysis of App Stores as a form of software repository mining. Information is readily available in the form of pricing and review data; Harman et al. [20] analysed this information and found a correlation between app rating and rank of downloads. Some studies have used open source apps in order to circumvent lack of available code: Syer et al. [25] compared Blackberry and Google's App Stores using feature equivalent apps, analysing source code, dependencies and code churn; Ruiz et al. [26] studied code reuse in Android apps; Minelli and Lanza produced a tool to mine and analyse apps [27]; Syer et al. [28] studied the effect of platform independence on source code quality.

Other studies extract and use API or other information from Android apps: Linares-Vásquez et al. [29][30] found that more popular apps use more stable APIs, and analysed API usage to detect energy greedy APIs; Gorla et al. [19] clustered apps by category to identify potentially malicious outliers in terms of API usage; Ruiz et al. [31] studied the effect of ad-libraries on rating; and Avdiienko et al. [32] used extracted data flow information to detect potentially malicious apps through abnormal data flow.

User reviews are a rich source of information concerning features users want to see, as well as bug and issue reports. They can serve as a communication channel between users and developers. Several studies have utilised (*F*)ully complete user review datasets, and these analyse many more user reviews than those using (*Pa*)rtially complete user review datasets: a mean of 9.8 million reviews are used by studies on *F* datasets, which starkly contrasts with a mean of 0.2 million reviews used by studies on *Pa* datasets. A summary of recent work on App Review Mining and Analysis can be found in Table II.

In 2012 Hoon et al. [11] and Vasa et al. [12] collected an *F* dataset containing 8.7 million reviews from the Apple App Store and analysed the reviews and vocabulary used.

Hoon et al. [1] then further analysed the reviews in 2013, finding that the majority of mobile apps reviews are short in length, and that rating and category influences the length of reviews. Another *F* sample was used in the 2013 study by Fu et al. [2], which analysed over 13 million Google Play reviews for summarisation.

Studies on *Pa* datasets use smaller sets of reviews, yet have produced useful and actionable findings. In 2013 Iacob and Harrison [7] presented an automated system for extracting and analysing app reviews in order to identify feature requests. The system is particularly useful because it offers a simple and intuitive approach to identifying requests. Iacob et al. [5] then studied how the price and rating of an app influence the type and amount of user feedback it receives through reviews. Khalid [3], [13] used a small sample of reviews in order to identify the main user complaint types in iOS apps. Pagano and Maalej [4] gathered a sample of 1.1 million reviews from the Apple App Store in order to provide an empirical summary of user reviewing behaviour, and Khalid et al. [9] studied the devices used to submit app reviews, in order to determine the optimal devices for testing.

Several authors have incorporated sentiment in their study of reviews. Galvis Carreño and Winbladh [8] extracted user requirements from comments using the ASUM model [33], a sentiment-aware topic model. In 2014 Chen et al. [10] produced a system for extracting the most informative reviews, placing weight on negative sentiment reviews. Guzman and Maalej [6] studied user sentiments towards app features from a small multi-store sample, which also distinguished differences of user sentiments in Google Play from Apple App Store.

Research on app reviews is recent at the time of writing, but much work has been done on online reviews (e.g. [21][34][35][36]), all built upon by the papers mentioned. Morstatter et al. [37] carried out a similar study to ours on Twitter data, finding that the subset of tweets available through the Twitter Streaming API is variable in its representation of the full set, available via the Twitter Firehose dataset.

VIII. CONCLUSIONS

Sampling bias presents a problem for research generalisability, and has the potential to affect results. We partition app and review data into three sets of varying completeness. The results from a Wilcoxon test between observable metrics such as price, rating and download rank between these partitions show that the sets differ significantly. We show that by appropriate data reduction of user reviews to a subset of user requests, we can learn important results through correlation analysis. For example, we find a strong inverse linear correlation between the prevalence of topics in apps and user requests. Our results suggest that user requests are more robust to the App Sampling Problem than apps, and that correlation analysis can ameliorate the effects of sample bias in the study of partially complete app review data. We build on the methods used by Iacob & Harrison [7] to extract requests from app reviews, in addition to using topic modelling to identify prevalent themes in apps and requests as a basis for analysis.

REFERENCES

[1] L. Hoon, R. Vasa, J.-G. Schneider, and J. Grundy, "An analysis of the mobile app review landscape: trends and implications," Faculty of Information and Communication Technologies, Swinburne University of Technology, Tech. Rep., 2013.

[2] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. New York, NY, USA: ACM, 2013, pp. 1276–1284.

[3] H. Khalid, "On identifying user complaints of iOS apps," in *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, 2013, pp. 1474–1476.

[4] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study." in *Proceedings of the 21st. IEEE International Requirements Engineering Conference (RE '13)*. IEEE, 2013.

[5] C. Iacob, V. Veerappa, and R. Harrison, "What are you complaining about?: A study of online reviews of mobile applications," in *Proceedings of the 27th International BCS Human Computer Interaction Conference (BCS-HCI '13)*. British Computer Society, 2013, pp. 29:1–29:6.

[6] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *22nd IEEE International Requirements Engineering Conference (RE '14)*, 2014.

[7] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*. IEEE Press, 2013, pp. 41–44.

[8] L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, 2013, pp. 582–591.

[9] 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 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE '14)*. New York, NY, USA: ACM, 2014, pp. 610–620.

[10] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "Ar-miner: Mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on Software Engineering (ICSE '14)*. New York, NY, USA: ACM, 2014, pp. 767–778.

[11] L. Hoon, R. Vasa, J.-G. Schneider, and K. Mouzakis, "A preliminary analysis of vocabulary in mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference (OzCHI '12)*. New York, NY, USA: ACM, 2012, pp. 245–248.

[12] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi, "A preliminary analysis of mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference (OzCHI '12)*. New York, NY, USA: ACM, 2012, pp. 241–244.

[13] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What do mobile app users complain about? A study on free iOS apps," *IEEE Software*, vol. 99, no. PrePrints, p. 1, 2014.

[14] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[15] Y. Bejamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal statistical Society (Series B)*, vol. 57, no. 1, pp. 289–300, 1995.

[16] A. Vargha and H. D. Delaney, "A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[17] Creative Research Systems. (2012) Sample size calculator. [Online]. Available: http://www.surveysystem.com/sscalc.htm

[18] Y. Yang, J. Stella Sun, and M. W. Berry, "APPIC: Finding The Hidden Scene Behind Description Files for Android Apps," Dept. of Electrical Engineering and Computer Science University of Tennessee, Tech. Rep., 2014.

[19] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *Proceedings of the 2014 International Conference on Software Engineering (ICSE '14)*. ACM Press, June 2014, pp. 292–302.

[20] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores." in *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR '12)*. IEEE, 2012, pp. 108–111.

[21] N. Jindal and B. Liu, "Opinion spam and analysis," in *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM '08)*. ACM, 2008, pp. 219–230.

[22] D. M. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[23] D. Newman, A. Asuncion, P. Smyth, and M. Welling, "Distributed algorithms for topic models," *Journal of Machine Learning Research*, vol. 10, pp. 1801–1828, Dec. 2009.

[24] Andrew, Kachites, and McCallum. MALLET: A Machine Learning for Language Toolkit. [Online]. Available: http://mallet.cs.umass.edu

[25] M. D. Syer, B. Adams, Y. Zou, and A. E. Hassan, "Exploring the development of micro-apps: A case study on the blackberry and android platforms," in *Proceedings of the 2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation (SCAM '11)*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 55–64.

[26] I. J. M. Ruiz, M. Nagappan, B. Adams, and A. E. Hassan, "Understanding reuse in the android market," in *ICPC*, 2012, pp. 113–122.

[27] R. Minelli and M. Lanza, "Samoa — a visual software analytics platform for mobile applications," in *Proceedings of ICSM 2013 (29th International Conference on Software Maintenance)*. IEEE CS Press, 2013, pp. 476–479.

[28] M. D. Syer, M. Nagappan, B. Adams, and A. E. Hassan, "Studying the relationship between source code quality and mobile platform dependence," *Software Quality Journal (SQJ)*, 2014, to appear.

[29] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "API change and fault proneness: A threat to the success of android apps," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013)*. New York, NY, USA: ACM, 2013, pp. 477–487.

[30] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Mining energy-greedy API usage patterns in android apps: An empirical study," in *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. ACM, 2014, pp. 2–11.

[31] I. J. Mojica, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "Impact of ad libraries on ratings of android mobile apps," *IEEE Software*, vol. 31, no. 6, pp. 86–92, 2014.

[32] V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, S. Arzt, S. Rasthofer, and E. Bodden, "Mining apps for abnormal usage of sensitive data," in *2015 International Conference on Software Engineering (ICSE)*, 2015, to appear.

[33] Y. Jo and A. H. Oh, "Aspect and sentiment unification model for online review analysis," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM '11)*. New York, NY, USA: ACM, 2011, pp. 815–824.

[34] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*. ACM, 2004, pp. 168–177.

[35] N. Jindal, B. Liu, and E.-P. Lim, "Finding unusual review patterns using unexpected rules," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. ACM, 2010, pp. 1549–1552.

[36] S. Hedegaard and J. G. Simonsen, "Extracting usability and user experience information from online user reviews," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, 2013, pp. 2089–2098.

[37] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the sample good enough? comparing data from twitter's streaming API with twitter's firehose," in *Proceedings of the Seventh International Conference on Weblogs and Social Media (ICWSM '13)*, 2013.