Research Note
RN/14/10

App Store Analysis: Mining App Stores for Relationships between Customer, Business and Technical Characteristics

5 September, 2014

Anthony Finkelstein, Mark Harman, Yue Jia,

William Martin, Federica Sarro and Yuanyuan Zhang

Abstract

This paper argues that App Store Analysis can be used to understand the rich interplay between app customers and their developers. We use data mining to extract price and popularity information and natural language processing and data mining to elicit each app's claimed features from the Blackberry App Store, revealing strong correlations between customer rating and popularity (rank of app downloads). We found evidence for a mild correlation between price and the number of features claimed for an app and also found that higher priced features tended to be lower rated by their users. We also found that free apps have significantly (*p-value* < 0.001) higher rating than non-free apps, with a moderately high effect size ($\hat{A}_{12} = 0.68$). We also provide initial evidence that extracted claimed features are meaningful to developers (precision = 0.71, recall = 0.77). All data from our experiments and analysis are made available on-line to support further analysis.

App Store Analysis: Mining App Stores for Relationships between Customer, Business and Technical Characteristics

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

1 Introduction

App stores provide a rich source of information about apps concerning their customer-, business- and technically- focussed attributes. Customer information is available concerning the ratings accorded to apps by the users who downloaded them. This provides both qualitative and quantitative data about the customer perception of the apps. Business information is available, giving the number (or rank) of downloads and also price of apps. Technical information is available in the descriptions of apps, but it is in free text format, so data mining is necessary to extract the technical details.

In this paper we mine the Blackberry World app store for data to support App Store Analysis. The technical information we mine is provided by the free text description of each app. We mine this using techniques inspired by work on mining natural language descriptions for technical information. In this way, our work resembles work on mining other forms of natural language product information [6]. Though there has been work on app store analysis [7], we believe that ours is the first paper to data mine and analyse app features and their relationship to non-technical information.

We find ourselves at a unique situation in software engineering research: in no previous software engineering development and deployment environment have software engineering researchers been able to access publicly available data that links all of these important attributes:

- The customers' opinions of software, in the form of the reviews they leave;
- The popularity of software, in the form of its rank and/or number of downloads;
- The price charged for software;
- The technical claims made by developers concerning the list of features offered by their software.

Of course, this information may not be complete or fully reliable: customers may, for various reasons, leave reviews that do not reflect their true opinion. Either intentionally or unintentionally, developers may not be entirely truthful about the technical claims made. Price information may only concern the price of the app, and may not include 'in app purchases' and other costs associated with using the app. Nevertheless, it is not unreasonable to hope that broad observations about whole classes of apps may still prove to be robust; the large number of apps on which such observations are based tends to support robustness.

It is important to note that we are extracting *claimed features*, though hereinafter we shall often refer to them simply as 'features' for brevity. That is, the feature information we extract reflects features that are present in the descriptions of apps, but they are not necessarily *present* in the app itself. We believe that this is an interesting aspect of our app store analysis: it gives us an opportunity to explore the relationship between claimed features and other app store data. Claimed features denote an interesting technical category in its own right. Whether or not there is a relationship between claimed features and features present in the app remains an interesting topic for another paper.

Specifically, in this paper, we are concerned with the correlation between the price, popularity and ratings accorded to apps by their users. We are also interested in the correlation between these three properties of the features of the apps. Correlation analysis allows us to address fundamental questions for any app store, such as:

1) Do apps that tend to get a higher rating also tend to be more popular?
2) Do apps that cost the customer more tend to get a lower rating?

- *A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro and Y. Zhang are with the Department of Computer Science, University College London, Malet Place, London, UK, WC1E 6BT*
  E-mail: a.finkelstein@cs.ucl.ac.uk, {mark.harman, yue.jia, w.martin, f.sarro, yuanyuan.zhang}@ucl.ac.uk

3) Do the extracted features enjoy any of the above correlations?

The primary contributions of this paper are:

1) We introduce the concept of Mining App Stores for business/technical and customer information [1]. It is important to note that there has been previous work analysing apps, for example app security [41], code reuse between apps [36] and dependence analysis [5]. Our primary conceptual contribution is to introduce the idea that App Stores can be mined for connected sets of data, allowing us to analyse the relationship between technical, customer, and business aspects of the market.
2) We study the distributions of prices and ratings over all apps. We found a very large number of zero-rated apps. We find that prices tend to be lower than \$5.00 for most apps, but there are frequency peaks at 'round number' prices, such as \$10 and \$20.
3) We present a procedure to mine feature information from app descriptions. Our approach uses natural language processing algorithms to extract likely feature descriptions as bitri-grams (i.e., 2-grams or 3-grams). We report the results of a simple 'sanity check' empirical study of 15 software developers, to investigate whether the extracted bitri-grams capture what the developers were considered to be meaningful features. Our results show the developers agreed that 16 out 19 mined bitri-grams were features, while this happened only in 3 out of 19 cases for the random bitri-grams. We also observed that, on average, the developers show high precision (0.71), recall (0.77) and f-measure (0.73), suggesting that they often classify the mined bitri-gram as feature and the random bitri-gram as non-feature. This provides some initial tentative evidence that the features we extract are meaningful to developers.
4) We empirically investigate the correlations between price, rating and popularity for free and non-free apps and their claimed features. For both we find evidence for a strong correlation between ratings and downloads; highly rated apps are more frequently downloaded, as one might expect. We find little evidence of correlations between price and either rating or popularity for apps, but we did find evidence for a mild inverse correlation between feature price points and median feature rating for the price point; customers tend to rate higher priced features less favourably than lower priced features. We also find that free apps have significantly (p-value < 0.001) higher rating than non-free apps, with a moderately high effect size ($\hat{A}_{12} = 0.68$), suggesting that users are not entirely insensitive to the pricing choices of developers.

The rest of the paper is organised as follows: Section 2 introduces the overall app analysis framework. Section 3 describes the metrics that capture the attributes of a feature. Section 4 presents the design of our empirical study, the results of which are analysed in Section 5. Section 6 discuss the limitations of the present study, while Section 7 describes other work related to ours. Section 8 concludes and presents directions for future work.

2 App Analysis Framework

Our approach to app store analysis consists of four phases shown in Figure 1. The first phase extracts raw data from the app store (in this case BlackBerry App World[2], though our approach can be applied to other app stores with suitable changes to the extraction front end).

In the second phase we parse the raw data extracted in the first phase to retrieve all the available attributes of each app relating to price, ratings and textual descriptions of the app itself. In the third phase we leverage on app descriptions to identify technical information, in particular, we use data mining to extract the features of apps from their textual descriptions. The final phase computes metrics on the technical, business and customer information extracted.

The rest of this section explains each step of our approach in more detail, while Section 3 presents the metrics we introduced to analyse the information mined.

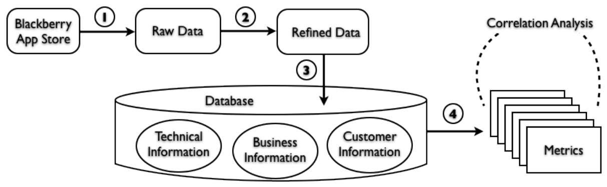


Fig. 1. **Overall App Analysis Architecture**: A four phase approach extracts, refines and stores app information for subsequent analysis.

Phase 1 (Data Extraction): We implemented a customised web crawler to collect raw webpage data from the Blackberry app store. Due to the existence of a large number of apps, the Blackberry app store does not provide a direct way to access all the apps iteratively. Thus, our crawler collects app data in two steps. First, it collects all category information from the app store and scans each category page to find the list of URLs of all the apps in each category. It then

1. Strictly speaking this claim attaches to our previous MSR 2012 paper [12]. This is a considerably extended version of that work, which develops the research agenda set out in the MSR paper.

2. http://appworld.blackberry.com/webstore/

visits the webpage of each app within each category and saves it as raw app data.

Phase 2 (Parsing): We extract a set of attributes for each app by parsing the raw data according to a set of search rules. The search rules are based on HTML tags identified manually, each of which specifies a unique signature for each attribute of interest. For example, we can retrieve the title of an app by searching the value of the $\langle h1 \rangle$ HTML tag with the attributes 'id=title' and 'class=awwsProductDetailsContentItemTitle'.

The extraction process cannot be entirely automated. Some attribute fields populated by humans require a further refinement process that accounts for the various ways in which the humans who populate the App Store data might provide equivalent information. For example, the values of the price field for a free app could be '0', 'Free', 'Free for one week' or a word that means 'free' in a language other than English. We assign a value 0 for the price of all such apps.

We developed search rules for the Blackberry apps to capture information about Name, Category, Icon, Description, Price, Release time, Version, Size, Language, Customers' Rating, Number of Ratings and Rank of Downloads. However, the analysis of this work is focused on the Category, Description, Price, Customers' Rating, and the Rank of Downloads attributes.

Once this manual step is complete the entire process is fully automated (until such time that the app store changes structure). To apply our approach to a different app store we need modify only the address information in the data extractor and the search rules in the parsing phase to accommodate the different app store structure and data representations, respectively.

Phase 3: (Data Mining Features): App features can be defined in many ways. For our purposes, feature information is data mined from app descriptions. For example, "7-days weather forecast" is a feature mined from apps in the weather category while "receive facebook message" is a feature mined from IM & Social Networking apps. The definition of an app feature (as mined by our process) is as follows:

"A feature is a claimed functionality offered by an app, captured by a set of collocated words in the app description and shared by a set of apps in the same category."

Since app descriptions are written in natural language, extracting features from the text requires data mining techniques usually associated with Natural Language Processing (NLP). We developed a simple four-step NLP algorithm to extract feature information and implemented it using the Natural Language Toolkit (NLTK), a comprehensive natural language processing package, written in Python [29].

Our feature extraction algorithm is presented in Algorithm 1. The first step extracts raw feature patterns, thereby identifying the 'coarse features' of apps. Feature patterns are informal patterns which developers used to list and clarify the features released. Figure 2 shows the description of a non-free Blackberry weather app, named "WeatherBug". We will use this example to illustrate our feature mining algorithm.

```
Stay informed and prepared with live local weather,
severe weather alerts, in-depth forecasts, camera views,
and radar maps. The features of the WeatherBug
application for the BlackBerry Storm include:

* Live neighborhood weather from over 8,000 weather
 stations in the U.S.
* Current weather, forecast and NWS alerts.
* 7-day and weekend forecasts.
* radar animation and cloud coverage.
* View snapshots and time-lapse animations from more
 than 2,000 weather cameras
* WeatherBug Community photos. Share you own weather
 photos.
```

Fig. 2. **WeatherBug**: An example of description of a weather app.

In Figure 2, the list starting with '*' is an example of a raw feature pattern which summarises the main features of the app. We locate raw feature patterns by searching HTML list in the description of apps. If the sentence prior to a HTML list contains at least one keyword from the set of words "include, new, latest, key, free, improved, download, option, feature", the HTML list is saved as the raw feature pattern for this app. We apply this process to all the apps in the same category to create a list of raw features, as shown in Figure 2.

Algorithm 1 Feature Extraction Algorithm

```
Require: apps
  rawFeatures = [ ]
  featureLets = [ ]
  for all apps do
     if featurePattern exists in currentApp.description then
        rawFeatures.append (extractFeaturePattern (currentApp))
     end if
  end for
  for all rawFeatures do
     refineRawFeatures (currentRawFeature)
  end for
  featureLets = findTriaGramCollocation (refineRawFeatures) {NLTK}
  features = getGreedyClusters (featureLets)
  return features
```

The second step of the algorithm refines the raw feature patterns by removing 'noise'. We first tokenise the raw feature patterns into a lower case token stream and then apply the following filtering: First, non-english and numerical characters are removed from the token stream. Secondly, incidental, unimportant 'noise' words are filtered out. The determination of what constitutes such an unimportant word is delegated to the English language STOPWORDS set in the NLTK data package. Finally, each remaining word is transformed into its 'lemma form' using the WORDNETLEMMATIZER function from NLTK, thereby homogenising singular/plural, gerund endings and other non-germane grammatical details. Ta-

TABLE 1
An example of a refined feature pattern.

$[live, neighborhood, weather, weather, station, us]$
$[current, weather, forecast, nws, alert]$
$[7\text{–}day, weekend, forecast, radar, animation, cloud, coverage]$
$[view, snapshot, time-lapse, animation, weather, camera]$
$[weatherbug, community, photos, share, weather, photo]$

TABLE 2
Featurelets: This table shows some examples of the featurelets extracted by applying the proposed approach to the weather app description reported in Figure 2.

Tri-gram collocated tokens	Tri-gram association score
$[animation, weather, camera]$	2891
$[neighborhood, weather, station]$	2826
$[share, weather, photo]$	2798
$[live, neighborhood, weather]$	2792
$[time\text{–}lapse, animation, weather]$	2780
$[7\text{–}day, weekend, forecast]$	2230

ble 1 shows an example of the refined feature pattern for the weather app example.

In the third step, the algorithm extracts a set of 'featurelets' from the refined feature patterns. A featurelet is a set of commonly occurring co-located words, describing a core function of apps. We perform a collocation analysis to find words that associate frequently from the refined feature pattern, built on top of NLTK's N-gramCollocationFinder package. We experimented with the settings for $N = [2, 3, 4]$ and found that the setting $N = 3$ generally achieves best results. The determination of 'best results' was made by the experimentors' subjective human assessment of whether the resulting n-grams appeared to be meaningful. However, this human judgement was more systematically tested in the simple 'sanity check' human study we performed to answer RQ4 (see Section 5.3).

Table 2 shows the featurelets extracted from the weather app example. Each of the featurelets on the left column has three tokens, because we used the tri-gram collocation model here. The right column shows the tri-gram association score, which indicates how frequently these tokens are associated together in the pool of the refined features of all weather apps. For each category of apps, we rank and select the best M featurelets based on the NLTK N-gram association measures. We experimented with the settings for $M = [100, 200, 500]$ and chose $M = 200$ in our experiments, once again based on the experimentors' assessment of the choice that produced the more apparently meaningful result.

Some extracted featurelets are similar to each other. For example, in Table 2, featurelets $[neighborhood, weather, station]$ and $[live, neighborhood, weather]$ share two common tokens ('neighborhood' and 'weather'). Step 4 applies a greedy hierarchical clustering algorithm to aggregate similar featureslets together, as shown in Algorithm 2. The algorithm treats each featurelet as one cluster initially. It then repeatedly combines clusters if their similarity measure is greater than a predefined similarity threshold.

The similarity measure is the number of common tokens shared by each cluster, and we chose 0.5 as the similarity threshold in our experiment, based on our assessment of result meaningfulness with different threshold values. The common words from each cluster are extracted as 'core features'. Table 3 shows the example of core features extracted from the featurelets showed in Table 2. We shall refer to a core feature as 'bitri-gram' since it can be represented by either a 2-gram or a 3-gram.

TABLE 3
Core Feature: This Table shows the core features (i.e., 'bitri-gram') extracted by applying the last step of the proposed approach to featurelets reported in Table 2

Core feature	Optional tokens
$[animation, weather]$	$[time\text{–}lapse, camera]$
$[neighborhood, weather]$	$[station, live]$
$[share, weather, photo]$	N/A
$[7\text{–}day, weekend, forecast]$	N/A

Phase 4: (Analysis): The final phase of our approach involves the analysis of the mined information. This phase is application specific. In the analyses presented in this paper, we collect metrics about apps and their features and use these in the correlation analysis and in the user study based on features. These metrics are defined in Section 3. The mined information could, of course, support many other analyses, and we make all our data available to support such subsequent downstream analyses[3].

3 Metrics for App Analysis

We introduce some simple metrics that capture the attributes of a feature, f in terms of the corresponding attributes of all apps that posses the feature f. This allows us to compute useful information about the features of an app. This section formalises the definitions of these metrics to support replication and future work.

We shall define our metrics with respect to an app database, which contains the information extracted for the app store. Let $AR(a, d)$, $AD(a, d)$ and $AP(a, d)$ denote the rating, rank of downloads and price, respectively, of the app a in the app database d. Let $\sharp(s)$ denote the size (cardinality) of set s. Let $S(f, d) = \{a_1, \ldots, a_m\}$ such that feature f is shared by all m apps $a_1, \ldots, a_m$ in an app database d.

We can extend $AR(a, d)$, $AD(a, d)$ and $AP(a, d)$ to the features extracted from app descriptions, by

3. Data from this paper is available at http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/UCLappA.html.

defining the rating, rank of downloads and price of a feature, f to be the average rating, downloads and price for all the apps that share f. More formally, we extend the metric X ($X \in \{AR, AD, AP\}$) defined from (app, database) pairs to numbers, to a metric F defined from (feature, database) pairs to numbers, as follows:

$$F(f,d) = \frac{\sum\limits_{a_i \in S(f,d)} X(a_i,d)}{\sharp(S(f,d))}$$

The same approach can be used to extend any metric X of type

$$\text{app} \times \text{database} \rightarrow \mathbb{R}$$

to one of type

$$\text{feature} \times \text{database} \rightarrow \mathbb{R}$$

Algorithm 2 Greedy Feature Cluster Algorithm

```
Require: featureLets
Require: greedyThreshold
  greedyClusters = [ ]
  greedySimilarities = [ ]
  for all featureLets do
    greedyClusters.add (featureLet)
  end for
  for i = 0 → len (greedyClusters) - 1 do
    currCluster = greedyClusters[i]
    for j = 0 → len (greedyClusters) - 1 do
      if  i == j  then
        currSimilairy = 0
      else
        currSimilairy = getSimilarity (currCluster, greedyClusters[j])
      end if
      greedySimilarities.add (currSimilairy)
    end for
    if max (greedySimilarites) > greedyThreshold then
      maxIndex = getMaxClusterIndex (greedySimilarites)
      mergeClusters (currCluster, greedyClusters [maxIndex])
    end if
  end for
  return  greedyClusters
```

4 The Design of the Empirical Study

This section explains the design of our empirical study, the research questions we set out to answer and the methods and statistical tests we used to answer these questions.

4.1 Research Questions

We are studying relationships between price, rating and popularity (rank of downloads) for apps and the features we extract from them. We therefore start by presenting baseline data on the distribution of these data.

Popularity is measured in terms of the rank of downloads, so this distribution is always a monotonically decreasing ranking. Also, since popularity is measured as a rank position in the league table of most downloaded apps (rank of downloads) this means that lower numbers (higher rank positions) indicate higher popularity on an ordinal scale.

From the app descriptions, we extracted 1,008 different features and so a natural question to ask is how these features distribute over the apps from which they are extracted. From the App Store, we extract rating and pricing information. We also present the distribution of these data, over both the apps and features extracted from these apps. These data form the answer to RQ0:

RQ0: Baseline data on Price, Rating and Feature Distributions

The next three research questions investigate the correlation between price, rating and popularity (i.e., rank of downloads) for non-free apps and between rating and popularity (i.e., rank of downloads) for free apps (those for which the price charged at the point of download is zero). These questions were addressed in the conference version [12] of this paper for non-free apps. In this journal extension of the conference paper, we also consider free apps, and investigate the correlations we find in greater depth.

RQ1: Price/Rating Correlation. What is the correlation between the Price (P) and the Rating (R) for non-free apps, overall and in each category?

RQ2: Price/Popularity Correlation. What is the correlation between the Price (P) and the rank of Downloads (D) for non-free apps, overall and in each category?

RQ3: Rating/Popularity Correlation. What is the correlation between the Rating (R) and the rank of Downloads (D) for free and non-free apps, overall and in each category?

In the conference version [12] of this paper, we observed correlation between rating and popularity rank of downloads), both for the apps themselves, and also for the features we extracted from them. In this extended version of the paper, we study this question in greater detail.

In the Blackberry App Store, at the time we took our snapshot, it was possible for a reviewer to assign a zero rating score. It is also possible that the particular app may have no reviews at all, which would also yield a zero rating score. It is not possible to distinguish between these two types of zero rated score. Furthermore, we might speculate that an app which has relatively few ratings available lacks sufficient evidence for the overall average rating recorded by users in general.

Therefore, we consider subsets of apps that have larger numbers of ratings available, and thereby enjoy a larger evidence base, from which we may draw inferences about average rating amongst the user community:

RQ3.1: Rating/Popularity Correlation over all reviews.

RQ3.2: Rating/Popularity Correlation over review subsets.

While the features we extract tend to have conceptually appealing names (i.e., bitri-grams), such as {near, wifi, hotspot}, we need to know whether they can be relied upon as meaningful technical items that a developer would consider to denote a feature.

In the conference version of this paper [12] we addressed this goal by analysing whether the statistical correlations observed at the app level can also be observed for the features we extract compared to the chance that such correlations could be observed for bitri-grams constructed entirely at random. In this paper we augment this previous study with a more human-based analysis. Specifically, we seek to investigate the degree to which human developers might believe that the bitri-grams extracted denote meaningful technical features:

RQ4: Do extracted bitri-grams denote meaningful features to developers?

Finally, in the conference version of the paper, we observed that there was no correlation between price and either rating or popularity. This surprised us, since we might conjecture that an app developer would have to try harder, *per se*, to garner higher ratings and popularity should they choose to charge a higher price.

Therefore, we investigate whether focussing on price ranges (rather than absolute price) might lead to higher correlation values. We also investigate whether there is a correlation between price and the number of features offered: perhaps apps that offer more features charge a higher price? This motivates our final research question, which investigates in more detail, the apparent absence of evidence for correlations involving price:

RQ5: Is there a stronger correlation involving Price when we 'zoom in' on specific ranges of price or between price and number of features or shared features in an app?

RQ5.1: Is there any difference in Rating and Popularity for free apps compared to non-free apps?

4.2 Data Employed in the Empirical Study

To answer the research questions, we constructed an app store database from the Blackberry store, taken by extracting information from all non-free apps present on the 1st of September 2011, our census date for this study. Table 4 shows summary data (i.e., number of apps, features, mean, median and minimum app price, rank of downloads, and rating) concerning the 19 categories in this app store database for non-free and free apps. The price is the price charged to download the app. The rating data is extracted from the reviews left by customers. The rank of downloads is the ranking position (relative to other apps) recorded by the app store for the downloads of the app at our census date.

We can observe that the number of apps contained in each category ranges from 45 to 11,504 for non-free apps and 42 to 1,257 for free apps. The categories 'Shopping' and 'News' contain the lowest number of non-free apps (i.e., 45 and 68, respectively), while the 'Weather' category contains the lowest number of free apps (i.e., 42). The categories 'Reference & Books' and 'News' contain the highest number of non-free and free apps, respectively.

4.3 Evaluation Criteria

We answer RQ0 by means of graphical analysis. In particular, we use histograms to visualise how many apps/features share the same price, rating, and how many apps share a same feature, over non-free and free apps.

To answer RQs 1-3 (i.e., investigate the correlation between price, rating and popularity of apps and features) we use scatterplots to show the relationship between two sets of data (i.e., price/rating, price/popularity, rating/popularity) and two association statistics (i.e., the Spearman's Rank Correlation and the Pearson Product-Moment Correlation) to measure their statistical dependence. The Spearman's correlation assesses how well the relationship between two pairs of observations can be described using a monotonic function, while the Pearson's correlation is a measure of their linear relationship. Both statistics range from +1 to - 1, where +1 indicates perfect correlation and -1 indicates a perfect inverse correlation. No correlation is indicated by 0.

To provide an in-depth analysis of the statistical correlation between price, rating and popularity (RQ3.2) we also analysed Spearman's (Pearson's) correlations at a finer grained level by grouping the apps depending on their minimum number of reviews (i.e., 0 to 9 reviews) and computing the correlation existing in each group. We visualise these results by means of graphs reporting, on the x axis, the number of reviews and, on the y axis, the rho and p-value provided by the Spearman's (Pearson's) test for each group. We refer to these graphs as correlation graphs.

To answer RQ4 we carried out a sanity check of our app feature mining technique, to see whether the feature descriptions extracted from this App Store are meaningful to humans. We asked the paper's co-authors (excluding the ones involved in the study preparation) and also software developers working in our other UCL research groups to fill in a simple on-line questionnaire[4]. The questionnaire contained both bitri-grams extracted by our feature mining technique and bitri-grams created by randomly selecting words from app descriptions. Each feature is presented to the users as a small set of collocated words (bitri-gram) describing a function shared by a set of apps in the same category. The users were given a list of 38 bitri-grams (half random, half mined) and the name of the category to which they belong.

4. The questionnaire is available at http://yuejia.cloudapp.net/appstore/questionnaire.

TABLE 4
Blackberry App World: Summary data computed for each category. The number of non-free apps and the mean price and rating are reported within the number of features mined for each category. Download information is provided by Blackberry App World as rank over all apps (free and non-free). To give a sense of the distributions of download rank positions, we present the mean, median and minimum ranks.

Category Name	Non-Free Apps							Free Apps					
	Number of	Number of	Price ($)	Rank of Downloads			Rating	Number of	Number of	Rank of Downloads			Rating
	Apps	Features	Mean	Mean	Median	Min	Mean	Apps	Features	Mean	Median	Min	Mean
Business	350	75	12.57	19063.48	18031	817	1.79	874	69	19491.55	16723	135	1.65
Education	576	58	5.68	22222.18	21739	1595	1.38	353	32	19024.98	15823	933	1.92
Entertainment	908	64	5.76	18413.13	16364	134	1.86	565	38	9721.01	6985	8	2.62
Finance	193	57	4.38	19593.59	16619	251	1.93	513	66	15394.49	12826	96	2.01
Games	2604	36	2.64	15919.49	13550.5	153	2.13	928	36	7109.65	5132.5	9	2.99
Health & Wellness	626	53	15.95	19852.88	18295.5	266	1.58	329	54	16168.94	13771	29	1.79
IM & Social Networking	150	55	4.42	14242.26	11513	22	2.55	305	60	8932.63	6171	0	2.29
Maps & Navigation	245	58	12.9	17140.75	13909	655	2.16	263	52	10489.11	7199	5	2.45
Music & Audio	499	60	2.05	24523.58	27248	204	0.99	1008	53	12699.36	10416.5	6	2.46
News	73	29	2.4	17485.36	15391	1393	1.73	1257	59	16037.91	13587	106	1.96
Photo & Video	393	72	2.51	21126.92	22879	15	1.4	192	50	7758.24	6064.5	33	2.47
Productivity	503	68	6.32	15124.95	11924	252	2.54	367	49	11050.12	8484	14	2.52
Reference & eBooks	11584	70	4.27	30388.93	31214.5	1155	0.12	366	22	18332.74	16800.5	16	1.67
Shopping	45	23	2.7	14785.51	11708	2543	2.33	229	41	13256.41	9933	56	2.08
Sports & Recreation	239	27	4.81	18808.38	16019	943	2.05	450	45	10509.9	8105.5	17	2.87
Themes	10936	32	3.12	21055.28	21254.5	18	1.68	552	29	5382.71	2836.5	27	3.35
Travel	764	57	4.81	25439.53	26112.5	553	0.67	441	56	13792.16	10828	12	2.38
Utilities	1362	66	4.61	16294.49	13994.5	63	2.32	950	59	11586.39	8932.5	7	2.67
Weather	58	48	7.51	12392.38	10288.5	309	2.44	42	39	7489.12	5193.5	21	2.61
All (avg)	1689.89	53.05	5.76	19151.21	17792.34	596.89	1.77	525.47	47.84	12327.76	9779.58	80.53	2.36

We set out to perform a 'sanity check' and not a full human-subject empirical study. Nevertheless, we did seek to minimise potential sources of bias, where we could: The same number of bi-grams and tri-grams were present in both sets. The developers were asked to say whether they believed that a given bitri-gram represented a feature or not. With the exception of two authors (who also participated) none of those polled had received any training nor any background on the research reported and none of those polled had previously seen any of the bitri-grams.

A total of 15 people filled in the questionnaire and their answers have been analysed by using both descriptive and accuracy measures. We use a histogram to show the users' agreement across features. To evaluate the classification accuracy we employed three common measures (i.e., Precision, Recall and F-measure) which are computed from the confusion matrix of Table 5. Precision allows us to assess the 'correctness' of the responses provided by a user (the degree to which a user agrees that our extracted feature bitri-grams were true features), while Recall allows us to measure the 'completeness' of user's responses (the degree to which they agreed with all of our bitri-grams). More formally:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

where TP is True Positives; FP is False Positives and FN is False Negatives.

To have an indication of a balance between correctness and completeness we computed the harmonic mean of Precision and Recall (i.e., F-measure), defined as follows:

$$F - measure = \frac{2TP}{2TP + FP + FN}$$

TABLE 5
The Confusion Matrix: Each column of the matrix represents the number of instances in a predicted class, while each row represents the instances in an actual class. A True Positive (Negative) is an instance that was true (false) and has been correctly classified as true (false), while a False Positive (Negative) is an instance that was true (false) and has been wrongly classified as false (true).

		Actual		
		True	False	Total
Predicted	True	TP	FP	$TP + FP$
	False	FN	TN	$FN + TN$
	Total	$TP + FN$	$FP + TN$	N

We do not claim that this study is a thorough full-scale empirical investigation of the question underlying RQ4. Such a study would require a paper in its own right. Our results, in answer to RQs 0, 1, 2 and 3 provide evidence that the bitri-grams we extract do capture *something* that carries meaning. In our answer to RQ4, we seek to investigate whether there is any, *prima facie*, evidence that this same *something* might be regarded as denoting 'a feature' by people with technical software development experience.

To answer RQ5 we investigated the possibility that there may be correlations between price/rating and price/download in sections of the data (perhaps for specific price ranges). We therefore further analysed these relationship by zooming into the scatterplots used to answer RQ3. We also considered the median rating and rank of downloads for each price point (for apps and features) to explore whether there is a correlation between these price points (chosen by developers) and the median rating (or popularity) given by customers for all apps (or features) charged at the associated price point. Moreover, we analysed whether there is any relationship between apps' price and their number of features or number of shared features by means of scatterplots and Spearman's and Pearson's correlation tests.

To answer RQ5.1 we investigated whether there is any statistically significant difference between the free and non-free apps. We also report on the effect size of any such significant differences, treating the apps for which we have data as a sample of all possible apps and using a non-parametric standardised effect-size measurement (Vargha and Delaney's $\hat{A}_{12}$ [1]) as a rough indicator of the degree of difference between the two types of app (free and non-free).

5 Result Analysis

5.1 RQ0. Baseline data on Price, Rating and Feature Distributions

Figure 3 shows the distributions of prices, ratings and features, over both apps and the features extracted from the apps.

There are fewer free-apps than non-free apps (see Figure 3(a)). The largest 'price point' (i.e., $> 10,000$ apps) is at \$0.99, dropping to approximately 5,000 priced at \$1.99 and thereinafter, the number of more expensive apps gradually decreases. Note that in this app store, prices are set at discrete dollar intervals (\$0.0, \$0.99, \$1.99, \$2.99 ...).

Despite the lower numbers of higher priced apps overall, we can observe peaks in the number of apps at the 'round number' price points (\$10, \$20 and \$30), though these prices are, more precisely \$9.99, \$19.99 and \$29.99 respectively. We observe a similar pattern for the prices of features extracted from the apps (see Figure 3(b)). Since the attributes of a feature (such as prices) are aggregates over all apps that share that feature, the averaging effect produces a more fine-grained distribution of possible price points.

There are also a large number of zero-priced features (i.e., 1,223). These are, by definition, features only contained in zero-priced apps. These features are not shown in the graph in Figure 3 since this column would be an outlier, thereby making the differentiation of other columns harder to read.

We found a very large number of zero-rated apps (see Figure 3(c)). Looking at the zoom-in subfigure for non-zero rated apps (Figure 3(d)), we observe that the majority of these apps (i.e., more than 2,500) are rated 4 or 5 star, about 2,000 apps are rated between 3 and 4 star, about 1,500 apps are rated between 2 and 3 star, and fewer than 1,000 apps are rated between 1 and 2 star.

The rating over features also reveals that a relatively high number (140 of the 1,008 features extracted) have a zero rating. These features, by definition, are only contained in apps that have a zero rating. They could be removed as being of little consequence, but we did not do this (or any other filtering of our algorithm's results), since we seek to validate our feature selection mechanism and do not want to bias these (or other) results by experimenter interference.

Since feature ratings are averaged over all apps that share the features, we see a finer-grained distribution of ratings for feature ratings than for app ratings, clustered around the original star scale. Not surprisingly, like the features' price distribution, this feature rating distribution is similar to the corresponding distribution for apps.

We report the number of features per app in Figure 3(g), which reveals that this distribution follows a power law: a very few (69 apps) app have more than 40 features, while a few (324 apps), have more than 20, while the vast majority (40,773 apps), have 10 or fewer features. In fact, more than a half of the apps (85%), have fewer than than 5 features. This is partly due to the fact that 65% of these apps belong to categories such as 'Themes' and 'Reference & eBooks', which provide users the sole functionality to download contents (e.g., a theme or a book) and partially due to the fact that there is no feature pattern in their descriptions. In the rightmost half of the graph, that does not include these categories, we find that 6,229 apps (14%) have more than 5 features.

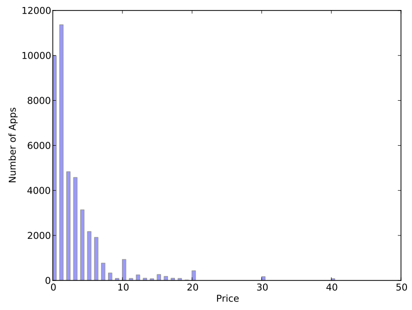
A manual inspection of the attributes of the 69 apps that had more than 40 features revealed that these apps were created by the same developers, have a similar description and share the same features that are related to photo editor and language dictionary functionality. This result reflects the earlier finding due to Ruiz et al. [36], that there is, in the Android app store, heavy code reuse in photography apps.

We also investigated the number of apps sharing the same feature. This also follows a power law as can be seen from Figure 3(f), which shows that a very few features are shared by more than one thousand apps. There are only 9 such highly prevalent features. A manual inspection revealed that these 9 features belong to apps from the 'Themes' category. They are features such as $[icon, set]$, $[home, screen, icon]$, $[background, screen]$.

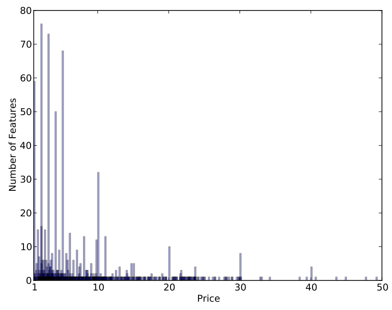
5.2 RQ1-3. Three correlations for non-free and free apps

Figure 4 shows the scatterplots between Price (P), Rating (R) and Rank of Downloads (D) at app and feature levels, respectively. The size of each point denotes the number of apps to which that data point refers.

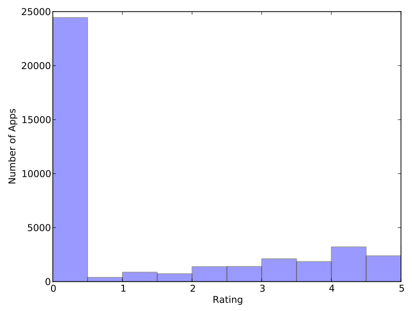
Graphs 4(a) and 4(b) suggest that the price of the apps is not strongly correlated with their popularity (i.e., apps of the same price can have different rank of downloads). However we can observe that the cheapest apps often have zero ratings. There is an



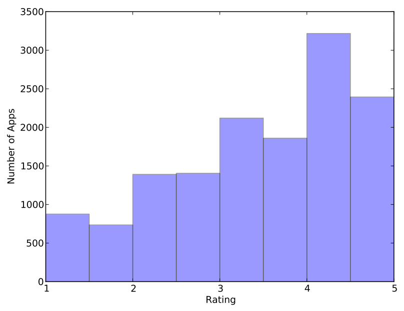
(a) Price distribution over apps



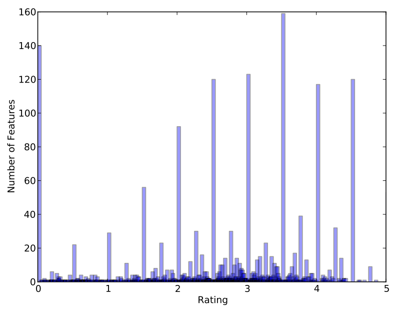
(b) Price distribution over features



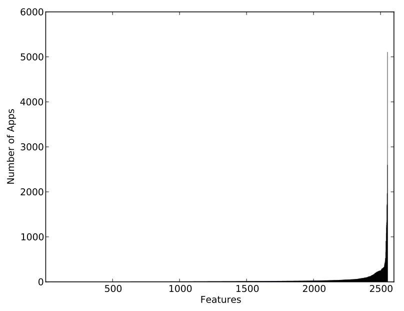
(c) Rating distribution over apps



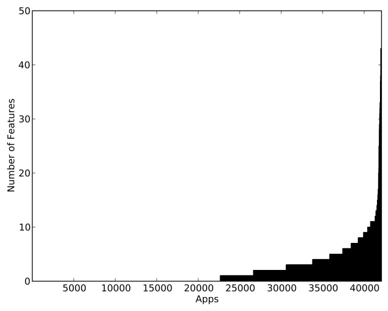
(d) Rating distribution over apps - zoom in



(e) Rating distribution over features



(f) Number of apps sharing a same feature



(g) Number of features per app

Fig. 3. **RQ0:** Distribution of price, rating and features at app and feature levels.

outlier in terms of price at $599; an app which has a price higher than that for many of the handsets on which it would reside when downloaded.

From graphs 4(c) and 4(d) we can observe that, regardless by their price, the non-rated apps tend to be less popular than the rated ones and that the higher the rating for an app, the more popular it tends to be.

Perhaps more importantly, when we look at the overall trend of the median values of rank of downloads for a given rating (Figures 4(e) and 4(f)), we can observe an apparently strong linear relationship for non-free apps. The relationship also appears to exist for free apps too, though it may have a slightly more exponential character. To investigate these observations for the scatter plots of rating scores to median rank of downloads, we calculated Spearman and Pearson correlation coefficients.

For non-free apps the Pearson correlation coefficient (rho) is 0.78 (p=0.004), and the Spearman correlation coefficient (rho) is 0.72 (p=0.013). For the free apps the Pearson correlation coefficient (rho) is 0.64 (p=0.033), and the Spearman correlation coefficient (rho) is 0.65 (p=0.037). This indicates that there is a strong correlation between rating and popularity for both free and non-free apps.

Since this finding has a potential impact (it is important to know that there is a relationship between rating and popularity) we investigated these correlations in some detail in the remainder of this section. We investigated various other model fits for the correlations, including polynomial and exponential. These next findings have to be treated with caution, since the 'rank of downloads' is an ordinal scale measurement and so a polynomial and logarithmic transformation may be considered questionable.

We report these results, with this caveat in mind, since they may shed further light on the relationship. The strongest Pearson correlation was found for the log of the median rank of downloads to the rating value. For non-free apps the Pearson rho is 0.78 (p=0.004). For free apps the Pearson rho is 0.66 (p=0.028).

In both cases, free and non-free, there is little difference between the results for the correlation for the log transformed rank of downloads and the original linear fit. However, we can also observe an interesting outlier for those apps with rating of 5.0 (the highest possible rating; five stars). Then rank of downloads for these apps is notably higher than the overall trend would suggest; five star apps seems to be peculiarly unpopular, on average. If we remove the outlier then all the correlations we have reported above increase notably.

For example, the Pearson rho for non-free apps becomes 0.91 (p=0.000) and for free apps it becomes 0.83 (p=0.003), while the log-transformed values indicate an even stronger exponential correlation: the Pearson rho is 0.96 for non-free and 0.93 for free apps (both with p=0.000). Without this outlier, the correlation for free apps seems to be more exponential than linear (so higher rated apps tend to be exponentially more popular), whereas for non-free apps the relationship appears to be linear whether or not we exclude the five star outlier.

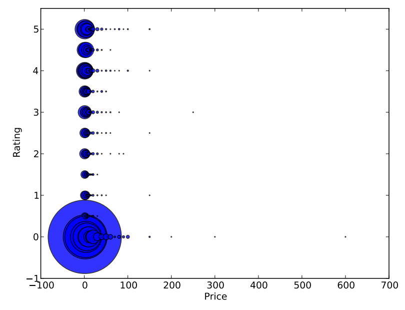
It is impossible to know exactly why the rating of five stars should be peculiar in this way. It would be tempting to speculate that there is something less reliable about five star ratings (particularly for apps that have only this top rating), even perhaps that a larger proportion of five star ratings might be suspicious than those at other rating levels. After all, if a developer were to rate their own app (or recruit others to do so) would that developer not wish for the highest possible rating? However, since correlation, on it own, cannot reveal causality, we leave this as an open question for further studies. Perhaps when we better understand how to assess the likely provenance of reviews, the question as to why five star ratings are peculiar can be answered more fully.

Similar observations about correlations between rating and poularity hold for the mined features. That is, there appears to be little correlation involving price (see Figures 4(i), 4(j)), while there is a strong (and apparently generally linear) correlation between rating and popularity: more highly rated features tend to be more popular (they have a lower rank of downloads). The correlation is far from perfect, overall, but the general linear trend is visually quite evident in Figures 4(k) and 4(l)).

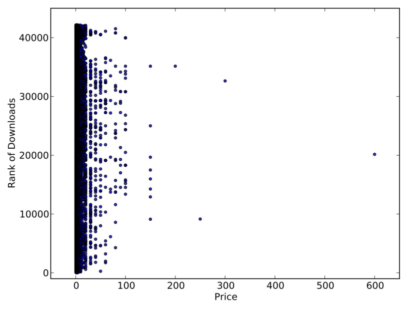
To provide a more quantitative assessment of these correlations for features and apps across, both within each category and overall report the Spearman and Pearson correlation values. Figures 5 and 6 show the Spearman's Rank and Pearson's correlation (solid line) and their significance (dashed line) obtained by grouping non-free and free apps, respectively, by their minimum number of reviews. In particular, we set the minimum number of reviews to range from 0 to 9 and plot the x axis as minimum number of reviews, and the y axis as the correlation (rho) value and p-value of the correlation test.

Space does not permit us to include all graphs of this form for all categories. However, all 38 graphs and other data are available at the paper's companion website[5]. We include these graphs to illustrate some general trends. We observed that there is an atypically higher correlation coefficient reported for the case where we include all apps (that is, we include all apps with zero or more ratings in the analysis) for both the Spearman and Pearson test. This higher correlation could be an artefact of the many apps with zero ratings; since these rating values are tied, by definition, this may tend to (artificially) inflate the

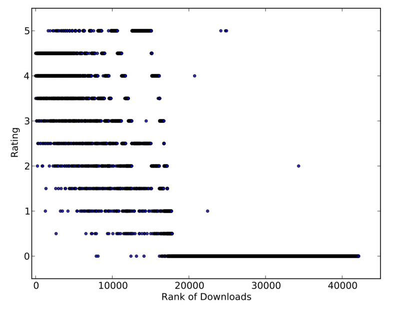
5. http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/resources/CorrelationGraphs.pdf.



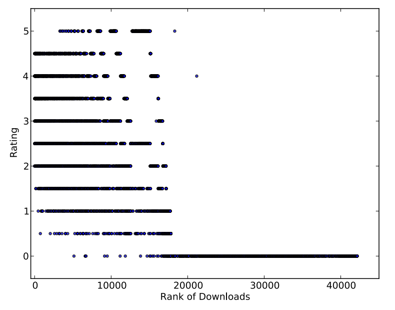
(a) PR non-free apps



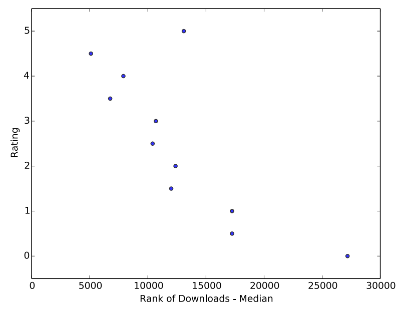
(b) PD non-free apps



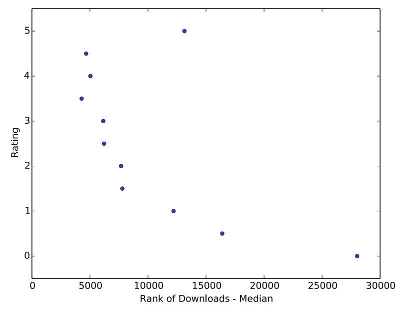
(c) RD non-free apps



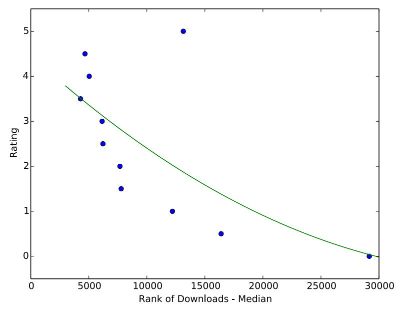
(d) RD free apps



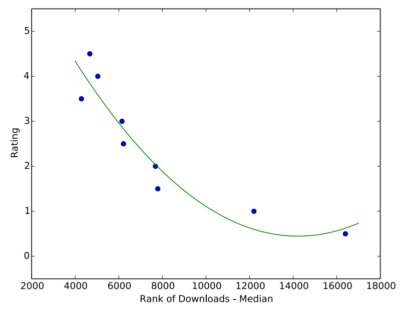
(e) MedianRD non-free apps



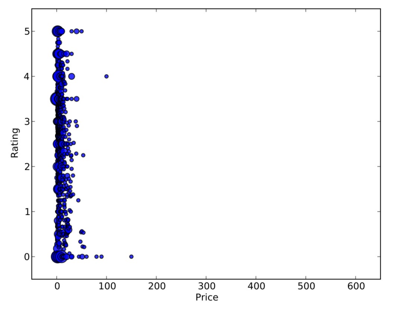
(f) MedianRD free apps



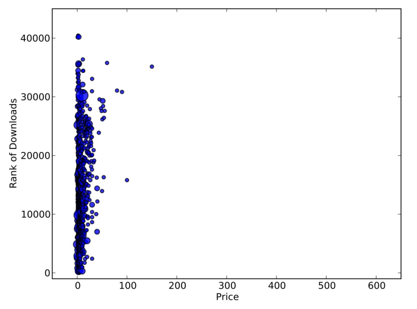
(g) MedianRD free apps - Polynomial model



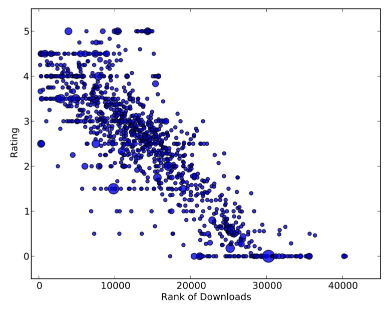
(h) MedianRD free apps ($0 < R < 5$) - Polynomial model



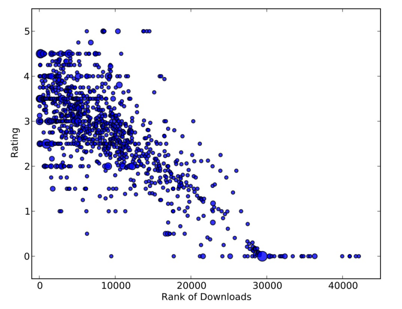
(i) PR non-free features



(j) PD non-free features



(k) RD non-free features



(l) RD free features

Fig. 4. **RQ1-3**: Scatterplot of Price (P), Rank of Downloads (D) and Rating (R) at app and feature levels.

correlation coefficient.

This was our motivation for additionally reporting on higher thresholds for the number of ratings required in order for the app to be included in the correlation analysis. As we move rightwards in these graphs, we reduce the number of apps considered, but increase the number of ratings required per app in order for the app to be included in the analysis. This reflects a trade off in the quality and quantity of evidence for customer rating.

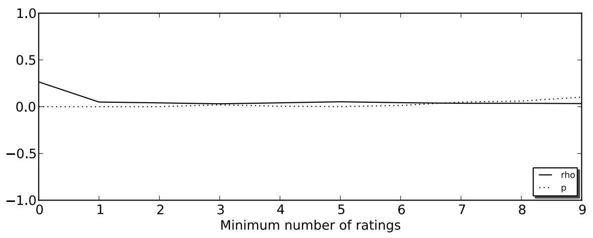
For correlation coefficients close to zero (no rank correlation) the amount of evidence needed is generally higher, in order for a reliable assessment of the correlation coefficient (rho) value. This is reflected by the change in the p value, which indicates insufficient evidence after $x = 2$ in the case of Figures 5(b) and 6(b). In order to be cautiously conservative about the correlations reported, we therefore based our claims that rest on qualitative analysis of correlation coefficients on analysis with 'rating filters' only up to a maximum of 2 (that is all apps with two or more ratings).

This quantitative analysis of Spearman and Pearson correlation coefficients can be found in Table 6 and Table 8 respectively. The decision as to when a correlation coefficient is sufficiently high that is reflects a degree of association is debatable. We report the coefficients so that the reader can make up their own mind. An absolute value for a correlation coefficient above 0.5 (with an associated p value less than 0.05) is, however, surely unlikely to arise by chance. Therefore, we treat this as a conservatively safe threshold above which we deem *some* correlation to exist in each case.

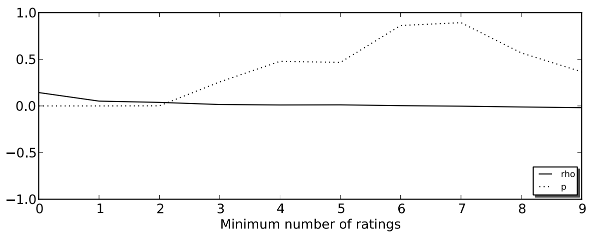
With this threshold in mind, we counted the number of correlation coefficients in each category, the absolute value of which was 0.5 or above. This count is reported in the final row of each table. As can be seen (from the columns labelled 'RD') in these tables, there are clearly many app and feature categories where there is a correlation between the rating and popularity (Rank of Downloads).

In particular, when the 'minimum number of reviews' threshold is set to zero (its most inclusive value), there is a correlation between rating and popularity for all but one category and in all but one case (18 out of 19) in three of the tables (and all cases for the fourth, concerning linear feature correlations). Of course, this value could be unduly influenced by tied ratings data (those apps with zero ratings). However, many strong correlations exist when we filter out all zero rated apps (in columns labelled 'MinReviews=1' and 'MinReviews=2').

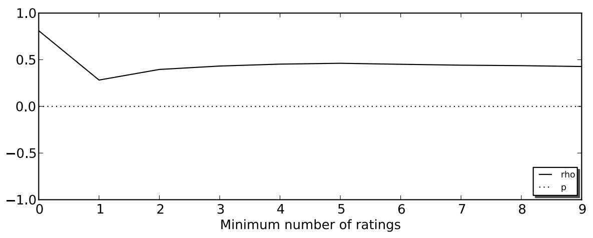
Perhaps somewhat surprisingly, we found little evidence for a correlation between *either* the price of an app and its rating, *or* between the price and the downloads of an app. This finding applies to both the app store as a whole and to almost all of the categories within it. This would suggest that, despite the plethora of apps and fierce competition, customers of non-free apps may not be as price sensitive as one might have thought; they tend to accord neither higher nor lower rating scores to more expensive non-free apps.



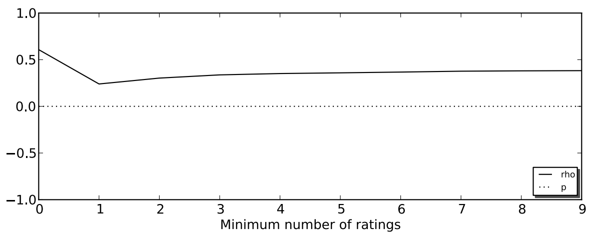
(a) Price VS Rating (non-free app)



(b) Price VS Rank of Downloads (non-free app)



(c) Rating VS Rank of Downloads (non-free app)

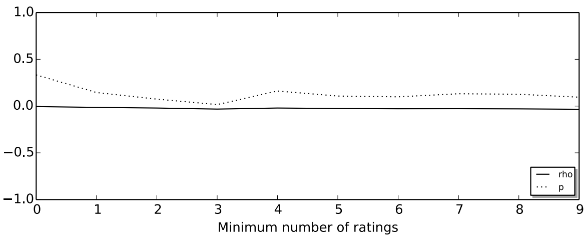


(d) Rating VS Rank of Downloads (free app)

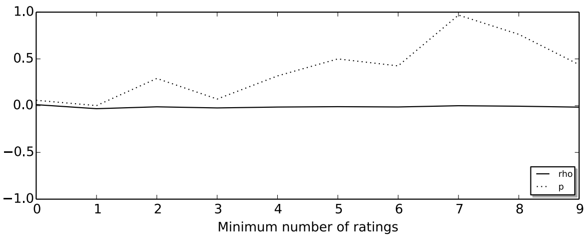
Fig. 5. **RQ1-3. Correlations Graphs**: The Figures show the Spearman Rank correlation values (solid line) and their significance (dashed line) obtained by grouping the apps by their minimum number of reviews.

Finally we observe that the Pearson's correlations between Rating and Rank of Downloads are stronger than the Spearman's ones suggesting that the relationship between these two variables has a more linear character than a monotonic one.

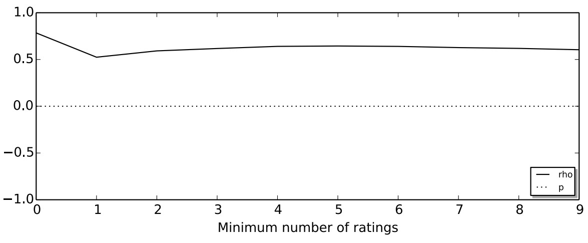
The correlations between rating and rank of downloads observed for apps can be also observed for the features we extracted, while no correlation has



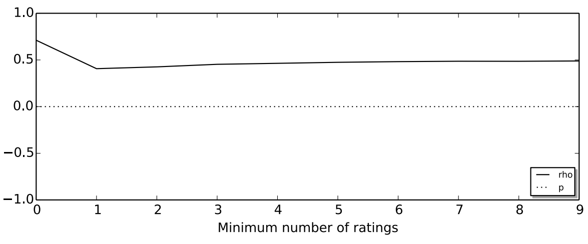
(a) Price VS Rating (non-free app)



(b) Price VS Rank of Downloads (non-free app)



(c) Rating VS Rank of Downloads (non-free app)



(d) Rating VS Rank of Downloads (free app)

Fig. 6. **RQ1-3. Correlations Graphs**: The Figures show the Pearson correlation values (solid line) and their significance (dashed line) obtained by grouping the apps by their minimum number of reviews.

been found between feature's price and rating. As can be seen from Table 7, we found strong correlations between the rating and the downloads for the features (as well as the apps) in almost every category (and also within the app store as a whole) when all the apps are considered (i.e., $MinRating = 0$). As we become more restrictive, the correlation values decrease in many categories for the same reason observed at the app level. Finally, the correlations observed for free features are, in general, lower than those observed for non-free features, perhaps suggesting that free features might be popular regardless of their rating.

In general, our results show that there is a correlation between customer rating and the rank of feature downloads and there is no correlation between feature price and feature downloads, nor between price and rating, replicating RQs1-3 at the feature level. This finding may offer useful guidance to developers in determining which features to consider when designing apps. As an example, they can provide insights into the added value of features under consideration for new products or next releases.

Thus, in answer to RQ1-3: our results show that there is a correlation between customer rating and the rank of app downloads for apps and the features extracted from them and for both free and non-free apps and features. However, there is very little evidence for any correlation between price and either rating or popularity.

5.3 RQ4. Do extracted bitri-grams denote meaningful features?

Table 10 reports the average True Positives, False Positives, True Negatives and False Negatives (TP, FP, TN, FN) obtained by the 15 software developers who participated in our 'sanity check' study. We presented the developers with a total of 38 bitri-grams (half mined, half randomly generated) in an arbitrary order, together with the corresponding category. We asked them to say if each bitri-gram represents a feature or not. Figure 7 shows the boxplots of the performance measures (i.e., Precision, Recall, F-measure) computed for the classification task carried out by the 15 developers. We observe that, on average, they achieved high Precision (0.71), Recall (0.77) and F-measure (0.73) values. This suggests that they often classify the mined bitri-gram as a feature and the random bitri-gram as a non-feature.

Figure 8 shows the number of developers that classified the mined and random bitri-grams as features. We clearly observe that the bitri-grams mined from the app descriptions by our algorithm were more often identified as features with respect to ones randomly extracted from the same descriptions. In other words, the agreement of the developers in identifying the mined bitri-grams as features is much higher than the agreement showed for the random ones. Indeed, in 16 out of 19 cases, more than 9 developers classified the mined bitri-grams as feature, while this happened in only 3 out of 19 cases for the random bitri-grams. Manually inspecting the three mined features that achieved the lowest agreement with the developers, we found that only one could be misleading, (i.e., $[number, time]$ in the 'Utility' category) while the other two are still meaningful in relationship to the corresponding category (i.e., $[activity, time]$ in the 'Business' category and $[automatically, centred]$ in the 'Maps & Navigation' category).

Thus, in answer to RQ4 we find that there is evidence that the bitri-grams of features extracted are meaningful to humans.

TABLE 6

Spearman Correlation Results for RQ1-3 at the App Level: The first 12 columns present the Spearman Rank correlation values computed for non-free apps, while the final 6 present the values we computed for free apps. We present the results obtained for each subset of apps having at least 0, 1, and 2 reviews. In all of these columns, the single letter labels stand for (P)rice, (R)ating and (D)ownloads.

Name of Categories	Non-Free Apps									Free Apps		
	MinReviews=0			MinReviews=1			MinReviews=2			MinReviews=0	MinReviews=1	MinReviews=2
	PR	PD	RD	PR	PD	RD	PR	PD	RD	RD	RD	RD
Business	0.02	0.03	0.83	-0.04	0.03	0.40	-0.01	0.07	0.52	0.81	0.40	0.46
Education	-0.10	-0.05	0.83	-0.06	0.04	0.52	-0.08	0.11	0.64	0.80	0.43	0.53
Entertainment	-0.17	-0.21	0.81	0.12	0.00	0.37	0.02	0.05	0.57	0.46	0.27	0.31
Finance	0.33	0.43	0.81	0.09	0.27	0.35	0.28	0.38	0.46	0.71	0.14	0.25
Games	-0.10	-0.01	0.76	-0.20	-0.03	0.42	-0.17	-0.05	0.49	0.27	0.16	0.23
Health& Wellness	-0.28	-0.26	0.85	-0.15	-0.06	0.52	-0.15	0.03	0.54	0.75	0.30	0.38
IM & Social Networking	-0.21	0.02	0.63	-0.30	0.10	0.28	-0.30	0.08	0.41	0.50	0.32	0.35
Maps & Navigation	-0.06	0.01	0.78	-0.04	0.15	0.45	-0.12	0.19	0.50	0.56	0.34	0.43
Music & Audio	0.42	0.33	0.76	-0.08	0.11	0.44	-0.20	-0.05	0.52	0.52	0.05	0.05
News	0.07	0.16	0.79	0.06	0.32	0.33	0.20	0.31	0.40	0.73	0.31	0.39
Photo& Video	0.02	0.06	0.82	-0.11	0.09	0.21	-0.09	-0.01	0.50	0.48	0.43	0.45
Productivity	0.01	0.08	0.73	0.02	0.14	0.35	0.00	0.12	0.37	0.58	0.37	0.45
Reference & eBooks	0.09	0.13	0.32	0.01	0.03	0.60	0.00	0.02	0.58	0.83	0.57	0.53
Shopping	0.26	0.21	0.67	0.12	0.03	0.28	-0.08	0.19	0.16	0.59	0.31	0.30
Sports & Recreation	-0.10	-0.02	0.77	-0.14	0.04	0.31	0.13	0.23	0.56	0.31	0.04	0.17
Themes	0.16	0.15	0.81	0.04	0.05	0.07	0.01	-0.02	0.17	0.04	-0.09	-0.09
Travel	0.04	-0.02	0.75	0.21	0.22	0.85	0.34	0.30	0.87	0.54	0.05	0.14
Utilities	-0.10	-0.03	0.77	-0.11	0.05	0.27	-0.12	0.00	0.45	0.55	0.29	0.37
Weather	0.07	0.12	0.54	0.19	0.21	-0.04	0.25	0.16	0.10	0.66	0.58	0.55
All	0.10	0.12	0.79	0.02	0.04	0.27	0.01	0.02	0.39	0.60	0.23	0.30
Some correlation	0	0	18	0	0	4	0	0	10	13	2	3

TABLE 7

Spearman Correlation Results for RQ1-3 at the Feature Level: The first 12 columns present the Spearman Rank correlation values we computed for non-free features, while the final 6 present the values we computed for free features. We present the results obtained for each subset of apps having at least 0, 1, and 2 reviews. In all of these columns, the single letter labels stand for (P)rice, (R)ating and (D)ownloads.

Name of Categories	Non-Free Features									Free Features		
	MinReviews=0			MinReviews=1			MinReviews=2			MinReviews=0	MinReviews=1	MinReviews=2
	PR	PD	RD	PR	PD	RD	PR	PD	RD	RD	RD	RD
Business	-0.36	-0.38	0.78	-0.32	-0.22	0.61	-0.10	-0.08	0.72	0.85	0.50	0.56
Education	-0.16	-0.27	0.87	0.28	-0.25	0.03	-0.05	-0.63	0.21	0.68	0.31	0.19
Entertainment	-0.30	0.05	0.57	-0.23	0.37	-0.07	0.00	0.31	0.09	0.18	0.03	0.04
Finance	0.12	0.28	0.46	-0.01	0.16	0.31	0.29	0.04	0.09	0.64	0.32	0.32
Games	-0.20	0.10	0.77	-0.15	0.16	0.25	-0.12	-0.13	0.59	0.36	0.21	0.09
Health& Wellness	-0.40	-0.50	0.93	-0.25	-0.37	0.68	-0.35	-0.10	0.69	0.87	0.63	0.57
IM & Social Networking	-0.36	-0.19	0.57	-0.24	-0.15	0.31	-0.21	-0.19	0.44	0.59	0.45	0.42
Maps & Navigation	0.48	0.42	0.90	0.35	0.28	0.79	0.28	0.29	0.88	0.58	0.23	0.28
Music & Audio	-0.05	0.00	0.74	-0.15	-0.12	0.49	-0.18	-0.01	0.65	0.13	-0.14	-0.24
News	0.12	0.05	0.75	0.17	-0.43	0.35	-0.05	-0.77	0.40	0.78	0.35	0.57
Photo& Video	-0.37	-0.30	0.80	-0.47	-0.26	0.47	-0.55	-0.47	0.60	0.28	0.28	0.28
Productivity	0.24	0.23	0.86	0.19	0.26	0.37	0.09	0.12	0.41	0.76	0.68	0.58
Reference & eBooks	-0.02	-0.39	0.74	0.49	-0.03	0.31	0.58	-0.28	0.30	0.33	0.11	0.24
Shopping	-0.17	-0.56	0.73	-0.20	-0.70	0.52	0.27	-0.59	0.01	0.78	0.75	0.76
Sports & Recreation	0.25	0.25	0.79	0.00	0.08	-0.02	0.02	0.37	0.26	0.35	-0.18	-0.09
Themes	0.32	0.00	0.80	0.15	-0.12	0.19	0.07	-0.06	0.32	0.35	0.05	-0.15
Travel	0.34	0.15	0.82	0.27	0.02	0.55	0.24	-0.06	0.51	0.64	0.12	0.23
Utilities	0.03	0.06	0.87	-0.26	0.01	0.56	-0.29	-0.18	0.68	0.73	0.55	0.61
Weather	0.11	-0.03	0.67	-0.01	-0.22	0.67	0.01	-0.28	0.72	0.60	0.60	0.60
All	-0.17	-0.19	0.81	-0.10	-0.21	0.37	-0.14	-0.23	0.44	0.64	0.33	0.33
Some correlation	0	2	18	0	1	7	2	3	9	12	6	7

TABLE 8
Pearson Correlation Results for RQ1-3 at the App Level: The first 12 columns present the Pearson correlation values computed for non-free apps, while the final 6 present the values we computed for free features. We present the results obtained for each subset of apps having at least 0, 1, and 2 reviews. In all of these columns, the single letter labels stand for (P)rice, (R)ating and (D)ownloads.

Name of Categories	Non-Free Apps									Free Apps		
	MinReviews=0			MinReviews=1			MinReviews=2			MinReviews=0	MinReviews=1	MinReviews=2
	PR	PD	RD	PR	PD	RD	PR	PD	RD	RD	RD	RD
Business	-0.09	-0.07	0.82	-0.13	-0.15	0.62	-0.05	-0.09	0.71	0.75	0.52	0.51
Education	-0.07	-0.08	0.76	0.09	0.05	0.55	0.06	0.13	0.72	0.78	0.54	0.55
Entertainment	-0.07	-0.17	0.77	0.19	-0.06	0.53	0.09	-0.02	0.65	0.61	0.38	0.38
Finance	0.17	0.13	0.83	0.11	0.04	0.61	0.12	0.26	0.64	0.73	0.3	0.39
Games	-0.09	-0.01	0.77	-0.18	-0.05	0.55	-0.15	-0.07	0.59	0.47	0.22	0.3
Health& Wellness	-0.27	-0.28	0.8	-0.06	-0.14	0.61	-0.12	-0.14	0.65	0.73	0.44	0.5
IM & Social Networking	-0.16	-0.18	0.74	-0.17	-0.15	0.51	-0.25	-0.26	0.6	0.63	0.42	0.43
Maps & Navigation	0.02	0.01	0.8	0	0.04	0.63	0	0.17	0.67	0.69	0.46	0.55
Music & Audio	0.23	0.26	0.8	-0.09	0.07	0.7	-0.22	-0.03	0.72	0.65	0.31	0.27
News	0.01	0.03	0.73	-0.07	-0.07	0.44	0.14	0.1	0.46	0.74	0.45	0.5
Photo& Video	0.1	0.13	0.85	-0.14	-0.09	0.44	-0.24	-0.22	0.61	0.54	0.46	0.5
Productivity	-0.03	-0.02	0.82	0.04	0.03	0.56	0.02	0.06	0.56	0.69	0.52	0.59
Reference & eBooks	0.1	0.15	0.42	0.05	0.01	0.62	0.05	0.01	0.65	0.82	0.69	0.65
Shopping	0.09	0.06	0.77	0.05	0.03	0.48	-0.13	0.17	0.35	0.65	0.44	0.39
Sports & Recreation	0.05	0.1	0.75	-0.04	0.03	0.54	0.07	0.14	0.64	0.59	0.23	0.29
Themes	0.12	0.1	0.81	0.04	0.02	0.42	0.03	-0.01	0.45	0.54	0.13	0.05
Travel	0.16	0.05	0.69	0.3	0.18	0.7	0.37	0.24	0.78	0.66	0.23	0.2
Utilities	-0.05	-0.09	0.82	0.02	-0.04	0.52	-0.06	-0.03	0.61	0.71	0.44	0.49
Weather	0.1	0.16	0.69	0.17	0.17	0.17	0.22	0.13	0.31	0.77	0.72	0.73
All	-0.01	0.01	0.78	-0.01	-0.03	0.52	-0.02	-0.01	0.59	0.71	0.41	0.43
Some correlation	0	0	18	0	0	14	0	0	15	18	5	9

TABLE 9
Pearson Correlation Results for RQ1-3 at the Feature Level: The first 12 columns present the Pearson correlation values computed for non-free features, while the final 6 present the values we computed for free features. We present the results obtained for each subset of apps having at least 0, 1, and 2 reviews. In all of these columns, the single letter labels stand for (P)rice, (R)ating and (D)ownloads.

Name of Categories	Non-Free Features									Free Features		
	MinReviews=0			MinReviews=1			MinReviews=2			MinReviews=0	MinReviews=1	MinReviews=2
	PR	PD	RD	PR	PD	RD	PR	PD	RD	RD	RD	RD
Business	-0.41	-0.48	0.76	-0.51	-0.65	0.67	-0.36	-0.60	0.66	0.85	0.71	0.58
Education	-0.08	-0.20	0.84	0.28	-0.22	-0.02	-0.13	-0.45	0.23	0.72	0.48	-0.02
Entertainment	-0.42	-0.27	0.74	0.01	-0.06	0.37	0.11	-0.07	0.32	0.28	0.14	0.19
Finance	0.15	0.30	0.73	-0.05	0.13	0.62	0.14	0.09	0.20	0.67	0.38	0.41
Games	0.05	0.21	0.76	-0.09	0.18	0.13	-0.32	-0.31	0.66	0.26	0.23	0.00
Health& Wellness	-0.36	-0.51	0.89	-0.04	-0.39	0.71	-0.43	-0.30	0.72	0.91	0.68	0.60
IM & Social Networking	-0.52	-0.39	0.67	-0.18	-0.04	0.30	-0.14	-0.10	0.48	0.56	0.41	0.36
Maps & Navigation	0.38	0.21	0.88	0.30	0.12	0.85	0.27	0.27	0.92	0.61	0.51	0.53
Music & Audio	-0.01	0.05	0.75	-0.12	-0.11	0.43	-0.22	0.10	0.57	0.44	0.10	0.00
News	0.10	0.16	0.73	-0.07	-0.24	0.49	-0.25	-0.77	0.54	0.72	0.36	0.60
Photo& Video	-0.30	-0.17	0.85	-0.43	-0.15	0.49	-0.59	-0.35	0.51	0.33	0.32	0.32
Productivity	0.35	0.29	0.89	0.25	0.23	0.42	0.21	0.13	0.54	0.78	0.66	0.48
Reference & eBooks	-0.19	-0.46	0.88	0.53	-0.19	0.28	0.50	-0.29	0.45	0.56	0.01	0.19
Shopping	-0.13	-0.50	0.79	-0.22	-0.67	0.71	0.39	-0.41	0.17	0.77	0.76	0.79
Sports & Recreation	0.00	0.11	0.78	-0.35	-0.25	0.23	-0.30	0.15	0.33	0.68	-0.19	-0.02
Themes	-0.02	-0.24	0.83	0.02	-0.12	0.27	-0.03	0.21	0.19	0.21	-0.01	-0.21
Travel	0.35	0.16	0.70	0.31	0.13	0.55	0.29	0.09	0.55	0.64	0.15	0.31
Utilities	0.02	-0.05	0.88	-0.06	-0.01	0.51	-0.23	-0.15	0.67	0.75	0.70	0.72
Weather	0.17	-0.07	0.70	0.09	-0.18	0.65	0.11	-0.23	0.69	0.82	0.79	0.79
All	-0.21	-0.26	0.83	-0.07	-0.27	0.52	-0.08	-0.27	0.58	0.75	0.47	0.45
Some correlation	1	2	19	2	2	8	2	2	12	14	7	7

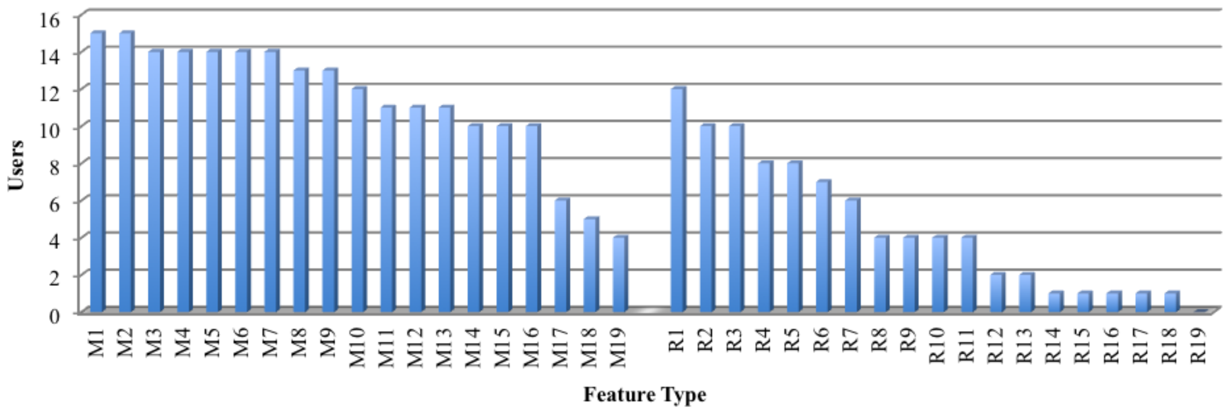
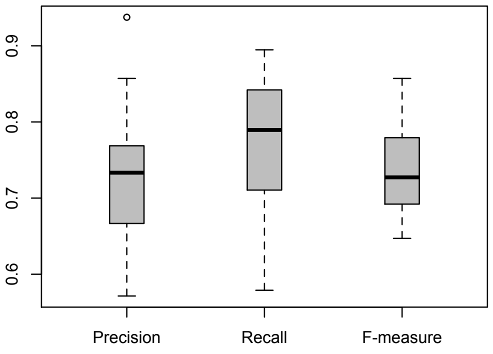


Fig. 8. **RQ4**: Number of developers that classified a bitri-gram as a feature. The x axis shows the bitri-gram type, i.e., Mined (M) or Random (R) and the y axis shows the number of users that classified it as a feature. We see that developers think, in general, that our mined bitri-grams are more 'feature like' than random bitri-gram from the same category description.

TABLE 10
RQ4-Confusion Matrix: average TP, FP, FN, TN values obtained by the 15 software developers involved in our feature's quality sanity check.

		Actual bitri-gram		
		Mined	Rand	Total
Predicted	Mined	16	9	19
	Rand	3	10	19
	Total	19	19	38

Fig. 7. **RQ4-Boxplots**: Precision, Recall and F-measure for the classification task carried out by the 15 software developers involved in the feature's quality sanity check.



5.4 RQ5. Is there a stronger correlation involving Price when we 'zoom in' on specific ranges of price or between price and number of features or shared features in an app?

The authors (and referees of an earlier version of this paper) were surprised that no evidence was found for correlations involving price. We found no evidence that price is correlated to either ratings or to popularity, neither for apps nor the features we extracted from them.

We considered the possibility that, though there is no overall correlation involving price, there may nevertheless, be correlations in sections of the data (perhaps for specific price ranges). We therefore further analysed the relationship between price/rating and price/download by zooming into the scatterplots shown in Figure 4. Moreover, we analysed whether there is any relationship between apps' price and their number of features or number of shared features.

From Figure 9 we can observe some interesting patterns:

1) Prices tend to be lower than \$5.00 for most apps, but there are frequency peaks at 'round number' prices, such as \$10 and \$20 (see Figures 9(a) and 9(b)). However, if we consider the median values (see Figure 9(c)) it is clear that there is no linear relationship between price and rank of downloads (i.e., Pearson rho = 0.165, p-value=0.385), while we can observe a mild rank correlation (i.e., Spearman rho=0.41, p-value=0.027).
2) The lower priced apps tend to have a higher rating (see Figures 9(d) and 9(e)). From the scatter plots we do see some evidence that the ratings accorded to apps priced below \$5.00 are slightly higher than those accorded to more expensive apps, but the correlation coefficient is extremely low: the Spearman rho=0.051, with a p-value = 0.000, while the Pearson rho=0.046, with a p-value = 0.000. Also, it should be noted that at this lower end of the price spectrum there are many tied values (e.g. all apps with price \$0.99) and this can artificially inflate the correlation values reported. If we look at the median values (see Figure 9(f)), we cannot find any significant correlations between price and rating (i.e., Pearson rho=-0.099, p-value=0.602 and Spearman rho=-0.159, p-value=0.401).
3) The more expensive apps tend to have more fea-

tures (see Figures 9(g) and 9(h)) and shared features (see Figures 9(i) and 9(j)). We found moderate correlations between price and median number of features (Pearson rho=0.46, p-value=0.007, Spearman rho=0.46, p-value=0.006) and between price and median number of shared features (Pearson rho $= 0.46$, p-value=0.007, Spearman rho $= 0.46$, p-value=0.006) when considering all apps (i.e., including those having zero features). The linear correlations between price and median number of features/shared features become stronger, while the Spearman's ones decreased dramatically (and, perhaps more importantly, lose their significance) when we consider only those apps having at least one feature (Pearson rho=0.54, p-value=0.001, Spearman rho=0.023, p-value=0.899) or at least one feature in common with other apps (Pearson rho $= 0.54$, p-value=0.001, Spearman rho=0.023, p-value=0.899).
This finding suggests that the apparent rank correlation for all apps (including those with no features at all) is a product of ties (the zero-featured apps have a tied number of features). However, the linear correlation is the one that is stronger so we conclude that there is overall evidence of a mild linear price to number-of-features correlation.

4) Though there is only the weakest evidence for any correlation between an app's price and either its rank of downloads or rating, there is stronger evidence for correlations between a feature's price and its rating (and also its rank of downloads). We investigated this further by computing correlation coefficients for the median rating and for the median rank of downloads per price point for all non-free features (see Figures 9(k) and 9(l)). For ratings, we found evidence of an inverse correlation between price and rating for both Pearson (rho $= -0.537, p = 0.000$) and Spearman (rho $= -0.559, p = 0.000$) correlations. For rank of downloads, the evidence was less strong: Pearson (rho $= -0.408, p = 0.000$) and Spearman (rho $= -0.422, p = 0.000$).

It is interesting to note that the correlation one might expect (higher prices are less likely to be favoured by users, surely?) is present with stronger evidence for the features than for the apps from which we extract these features. This could be because there are many more different price points and rating values for features (since feature properties are computed as averages over the apps that share the features). However, the strong correlation found is further evidence that the features we extract carry some meaning and that this meaning could be useful to developers.

In answer to RQ5 for apps, we found that there is a moderate correlation between apps price and median number of (shared) features. The higher the price the more features are claimed to be provided. However, the answer for features provides stronger evidence of an inverse correlation between the price rating; more expensive features tend to be less highly ranked.

5.5 RQ5.1: Is there any difference in Rating and Popularity for free apps compared to non-free apps?

From Table 4 we can observe that, on average, free apps have a lower rank of downloads than non-free apps (suggesting that, in general, free apps are more popular). We found that this difference is statistically significant according to the non-parametric Mann-Whitney 'U' Test (p-value < 0.001), with a notable effect size (the Vargha-Delaney normalised non-parametric effect size $\hat{A}_{12}$ is 0.76). The same observation holds for free features (i.e., free features are more popular than non-free ones, p-value < 0.001 and $\hat{A}_{12} = 0.70$).

From Table 4 we also observe that free apps provide the users slightly fewer features (see Table 4) on average, than their non-free counterparts. However, we found that this difference is not statistically significant according to the non-parametric Mann-Whitney 'U' Test (p-value= 0.847, $\hat{A}_{12} = 0.50$) .

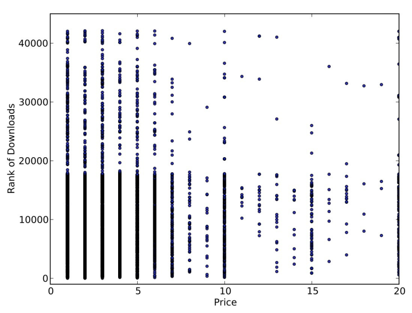
As for the rating, we can observe that the most highly rated non-free apps reside in the categories 'IM & Social Networking', 'Weather' and 'Productivity', while 'Themes' and 'Games' contain the most highly rated free apps. In general, we observe that free apps enjoy a higher rating, on average, compared to the non-free apps that reside in the same category (see Table 4). This difference is statistically significant (p-value < 0.001), according to the non-parametric Mann-Whitney 'U' Test and has a reasonably large effect size ($\hat{A}_{12}$ is $= 0.68$).

Overall, we find that there is strong evidence that the free apps are, in general, more popular than non-free apps and that they also enjoy higher ratings.

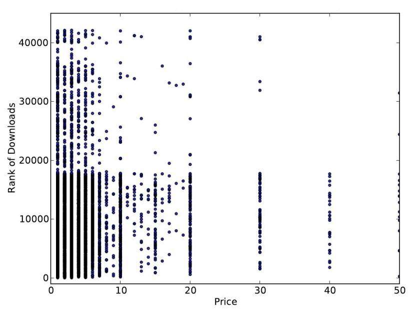
6 Threats to Validity

In this section we discuss the validity of our study based on three types of threats, namely *construct*, *conclusion*, and *external* validity.

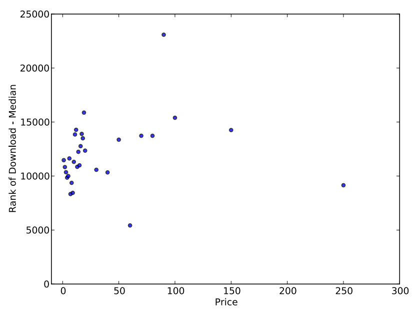
Construct validity concerns the methodology employed to construct the experiment. Our data is extracted from the Blackberry App Store. As such we are relying on the maintainers of the App Store for the reliability of our raw data. Undoubtedly, there will be inaccuracies and imprecisions in the data and these may have affected some of our computed data. In order to protect against possibly incorrect conclusions that may be drawn from such analysis, we have been careful to base all of our primary observations



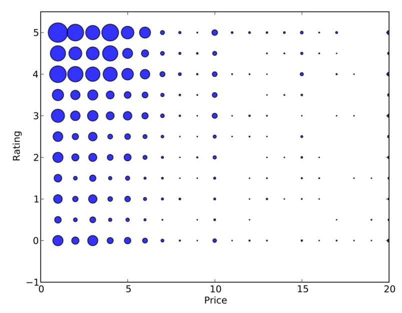
(a) Price VS Rank of Downloads



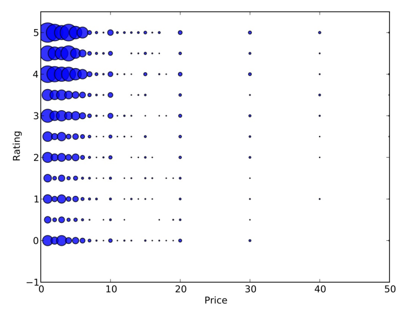
(b) Price VS Rank of Downloads



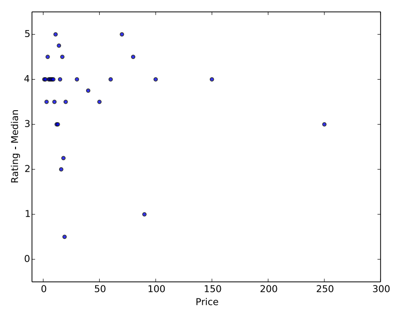
(c) Price VS Rank of Downloads (median)



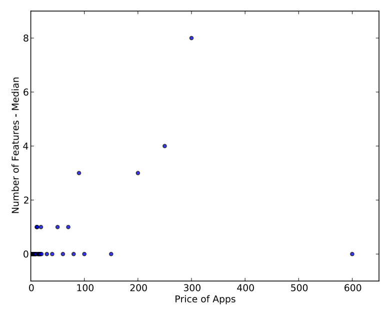
(d) Price VS Rating



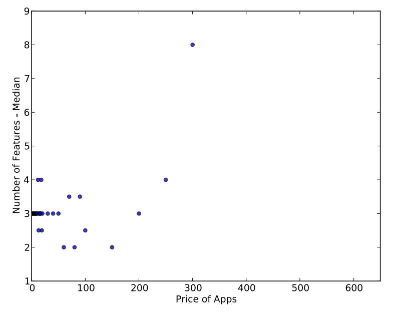
(e) Price VS Rating



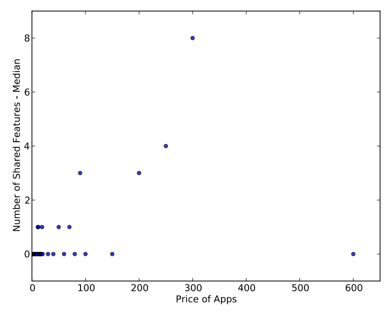
(f) Price VS Rating (median)



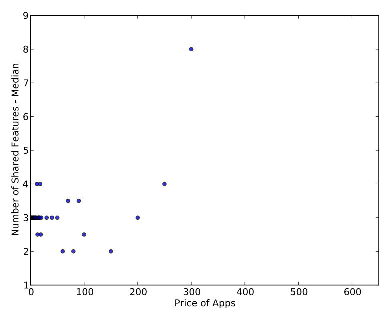
(g) Price VS Feature (median), $f \geq 0$



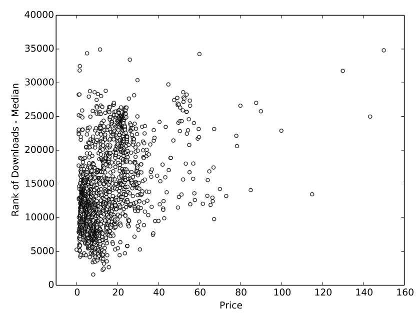
(h) Price VS Feature (median), $f > 0$



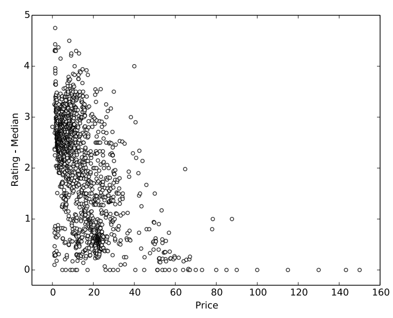
(i) Price VS Shared Feature (median), $f \geq 0$



(j) Price VS Shared Feature (median), $f > 0$



(k) Rank of Downloads - Median per Price point (features)



(l) Rating - Median per Price point (features)

Fig. 9. **RQ5**: Scatterplots of Price VS Rank of Downloads, Rating, Features and Shared Feature at different levels of granularity.

on analyses over large sets of data. By focusing on such 'macro level' statistical observations (rather than fine-grained detailed observations), we hope that our findings will prove to be robust in the presence of any inaccuracies and imprecision in the raw data.

Conclusion validity threats concern the issues that affect the ability to draw a correct conclusion. We carefully applied the statistical tests, verifying all the assumptions each inferential test requires concerning the distributions to which it is applied. We also reported widely-used measures to obtain a quantitative evaluation of the human study (i.e., Precision, Recall and F-measure). We also verified that the Matthews Correlation Coefficient was always positive, an important sanity check for any categorisation model [40]. Conclusion validity could also be affected by the selection and number of participants in the human study. To mitigate this threat, we plan to conduct replications on a larger number of participants.

Our approach to external threats is relatively standard for the empirical software engineering literature. That is, while we were able to obtain a set of categories that had a degree of diversity in application type and size, we cannot claim that our results generalise beyond the subjects studied. The number of features analysed in the human study could also threaten the external validity of the results. The rationale for selecting a relatively small number of features (i.e., 38) arose from the need to mitigate the potential problem that subjects may choose to drop out of the experiment (i.e., mortality).

A potential threat to generalisabily lies in our extraction of feature information from descriptions. Naturally, we do not claim that these extracted features include all the real features of the app. Indeed, we do not *even* claim that *any* of features we extract can be found in the app. Rather, we claim that there is evidence that what we have extracted tends to be meaningful feature descriptions (as indicated by our human sanity check) and that they denote features *claimed* to be included in the apps (according to the developers' own descriptions). Great care is required in extending our findings from 'claimed features' to features that are truly available to users of the app. Such extrapolation of our findings is not valid unless future work demonstrates a strong correlation between claimed and actual features.

7 Related Work

Much of the previous work on Mining Software Repositories (MSR) has tended to focus on understanding, predicting and, ultimately guiding the process of software evolution [14], [49]. The goal of App Store Analysis is to combine technical data with non-technical data such as user and business data to understand their inter-relationships. The number and granularity of the software products considered differs from previous work on mining non-app software: MSR typically uses a white box analysis of multiple applications [9] of software products of (sometimes) very large size [39].

By contrast, to mine app stores, we can use white box techniques where the source code of apps is available. However, we may also use a black box analysis of the apps, where source code is unavailable. As we have demonstrated in this paper, technical information can be extracted from sources other than the code of the app itself. We are also likely to consider potentially many more software products, but of perhaps smaller size, at least for the apps available at the time of writing (they may grow in size and complexity in future, as all software generally tends to do [25]).

Several other authors have also commented on general properties of App Store Analysis and its relationship to traditional software repository mining. For example, Syer et al. [43] seek to understand the differences in characteristics between apps and more conventional applications, drawing parallels between apps and UNIX utilities, while Nagappan et al. [33] and Menzies [30] discuss challenges and opportunities in app analysis.

Minelli and Lanza [32] also compared apps with traditional software systems, finding that apps are smaller and simpler (consisting of approximately 5.6k Lines of code, on average). However, they claim (and we agree) that this may be a transient effect, due to disappear as apps become larger and more complex.

Ruiz et al. [36] analysed Android code reuse, finding it to be prevalent compared to non-Android open source software. They also found that developers reuse software through inheritance, libraries and frameworks (a result we partly replicated for the Blackberry world app store in Section 5.1).

Yamakami [47] proposes the notion of 'open source based mobile platform software engineering' and discusses a business model involving software vendors, carriers and handset vendors. Want [46] highlights how the app store allows developers to integrate, test and distribute the key aspects of pervasive computing research.

Our results are based on a smartphone app store (though there is no reason to assume that they may not apply to other app stores). The first smartphone (named Simon) was designed as a concept product by IBM in 1992 [38]. Since then, there has been a dramatic increase in terms of the number of functionalities and platforms for the smartphones. The Apple App Store [19], opened in July 2008, contained over 775,000 apps by January 2013, while over 40 billion apps were reportedly downloaded in its first four years of operation [20]. The Google Play App Store (formerly named 'Android Market') [18] was opened in October 2008. There were 700,000 apps available with over 25 billion downloads as of September 2012 [17]. BlackBerry App World [16] opened in April

2009. There were more than 99,500 available apps in the store after three years of operation [21]. Clearly, there are bigger app stores than the Blackberry App World studied in this paper. We plan to extend our work to other app stores in future work. However, the results presented here for Blackberry concern an app store that is worth several hundreds of millions of dollars, so the potential monetary impact of the findings remains considerable.

Until relatively recently, there has been little work on app stores as a source of software engineering data. However, since 2012 there has been an explosion of research in this area. In the remainder of this section we briefly review this related work on App Store Analysis and its relationship with our work.

Much of the recent work on App Store analysis seeks to understand users' preferences as expressed through the reviews that they leave on the app store [10], [15], [22], [23], [24], [34], [37].

Iacob and Harrison [22] introduce a tool called MARA (Mobile App Review Analyzer) to extract feature requests from online reviews to assist app developers. Of 136,998 reviews analysed, the authors report that about a quarter (23.3%) are feature requests.

Pagano and Maalej [34] analysed over a million reviews from the Apple App Store, finding that most reviews are uploaded immediately after a new app release. They also found that feedback content has an impact on download numbers: positive messages usually lead to better ratings and vice versa. While they found that application ratings and ranks are not statistically independent, they did not find a statistical correlation, as we have found.

Khalid et al. [23], [24] manually analysed 6,390 reviews from a sample of 20 Apple App Store apps, reporting that the most common user complaints revolve around functional errors and crashes, as one might expect, but also among the most prevalent 'complaints' are requests for new features.

Hoon et al. [15] mined data from 8 million app reviews residing in the Apple App Store. Their analysis reveals that the vast majority of reviews are very short and that this brevity tends to increase with app age. They also reported a tendency for approximately half of the apps to decrease in average user rating over time.

Ruiz et al. [37] studied the prevalence and impact of advertisements in apps, which are an increasingly popular source of revenue for the app developer. They mined data from 519,739 versions of 236,245 different Android apps over 27 Google Play categeories. Perhaps surprisingly, even though apps may contain up to 28 different advertisement libraries, the authors found no evidence that the number of advertisements has an impact on app ratings, overall. However, the authors report that inclusion of certain advertisement libraries does appear to be negatively correlated with ratings.

Guzman and Maalej [10] perform a sentiment analysis on app reviews to help developers cope with the large number of reviews that may be available for their apps. They use topic modelling to extract and group together features mentioned in these reviews and report the results of an empirical validation with nine human subjects (including the two authors) on 2,800 reviews of 7 apps from the Google. Guzman and Maalej report that their approach has precision and recall of 0.59 and 0.51 respectively, according to their validation. These precision and recall values are lower than those found with our 'sanity check' human study of the features we extract. However, it should be noted that Guzman and Maalej are attempting the challenging task of imputing sentiment to reviewers as well as feature extraction from the reviews they study.

In this paper, we have shown that useful information can be extracted from descriptions. This is important, particularly in situations where no source code is available for an app. However, in many cases, there may be source code available and so any analysis performed about descriptions can also be augmented by whitebox study of the source code itself. Such white box and analysis is, of course, also an instance of App Store mining.

Gorla et al. [8] use API calls as a convenient way to understand the semantic behaviour of a large number of apps, the source code which they mine. They show how anomalous API calls can be used to detect aberrant or otherwise suspicious behaviour. Taba et al. [44] study 1,292 free Android apps from 8 app categories, reporting that users award significantly higher ratings to apps with simpler user interfaces. Linares-Vasquez et al. [28] study clones in 24,379 free Android apps, observing that developers' use of obfuscation techniques to protect their intellectual property has a tendency to increase false positive clone detection. Syer et al. [42] investigate the relationship between defects found in an app and the degree to which it is platform dependent (assessed in terms of API calls). They report a positive correlation between the number of defects found and platform dependence in the Android apps. Addressing this, Linares-Vasquez [27] seeks to support Android app maintenance in the presence of API and platform updates. While Syer et al. consider dependence within an app, Angeren et al. [45] investigate dependence between various attributes of apps in the App Store itself to give a perspective on the App Store as a software ecosystem [26].

In this paper, we have extracted feature information from app descriptions. Our approach is independent of whether these descriptions are truthful or not, since we merely seek to determine correlations between the ratings, popularity and prices accorded to these extracted claimed features. Nevertheless, the information we find will have additional potential applications for those descriptions that tend to be more truthful. Pandita et al. [35] introduce a tool WHYPER, which compares the permissions requested by the app and the app description. This allows them to highlight apps with suspect descriptions. Suspicion arises when mismatches are found between an app's technical declaration of permissions sought and its public declaration of features it offers. They used First Order Logic to analyze the sentences in the description. Yang et al., [48] also considered this problem, introducing an approach they call APPIC to compare features extracted from descriptions (using topic modelling) with the permissions declared for an Android app.

Lulu and Kuflik [3] cluster apps to help users retrieve the apps they have installed on their device. Their approach is also based on information extracted from the app description, but augmented by content from 'professional blogs'.

In many areas of software engineering research, it is some considerable time before widely available tooling catches up with research frontiers. It is therefore perhaps a sign of the rapidly increasing maturity of App Store Analysis that there are already several tools available for mining and extracting information from app stores. For example, as well as our own tool, APPIC, MARA and WHYPER already discussed above, Chen et al. [4] introduce AR-miner for extracting review information, Minelli and Lanza [31] introduce SAMOA, for visualising results of mobile app analysis, and Bakar and Mahmud [2] introduce OSSGrab, for extracting information from app stores.

Our work [12] was the first to argue that App Store Analysis can be used to understand the relationships between technical, business and social aspects of app stores. It was also the first to propose the incorporation of technical information (such as feature information, mined from app descriptions) as part of this analysis process. The present paper extends this initial analysis of non-free Blackberry apps, to consider both free and non-free apps and the correlations between their claimed features, rating, popularity and price in more detail.

8 Conclusions and Future Work

App stores provide a software development space and market place that are fundamentally different from those to which we have become accustomed for traditional software development: the granularity is finer and there is a far greater source of information available for research and analysis. Information is available on price, customer rating and, through the data mining approach presented in this paper, the features claimed by app developers. These attributes make app stores ideal for empirical software engineering analysis.

We have introduced a method to extract, from app store descriptions, usable information about the features of apps that captures some of the technical aspects of the apps in the store. We evaluated our approach on both the free and the non-free apps in the Blackberry App Store. We performed a simple empirical study as a sanity check that the feature information extracted was meaningful to developers. The study found high precision (0.71), recall (0.77) and f-measure (0.73), suggesting that the bitri-grams we extract to capture features are meaningful as feature descriptions for developers.

We found that the number of features per app (and the number of shared features between apps) follow a power law. We also found that, though there are a large number of zero-rated apps. The non-zero ratings accorded to apps by their users are, generally speaking, positive; more ratings occupy the higher, more favourable end of the rating spectrum.

The degree of correlation between rating, price and popularity is different for different app categories, as one might expect and as we report in detail in the paper. Our analysis indicates that there is a strong overall correlation between the ratings given to apps by their users and their popularity (i.e., rank of downloads). This correlation was observed for both free and non-free apps. This correlation is also present in the features we extract and so this feature information may be useful in its own right. We found that free apps received significantly higher ratings than their non-free siblings and that there is a mild correlation between price and the number of features offered, but we found little evidence for any correlation between the price of a non-free app and either its rating or popularity.

There are many potential avenues for future work that result from our findings. For example, we could use feature level clustering to re-draw and re-consider the boundaries of the categories of apps in an App Store, which may help to identify outliers that may have been mis-categorised. We can also study migration of features between categories over different snapshots of the app store.

In future, we also intend to investigate predictive models of customer evaluations, and the interplay between functional and non-functional properties of apps, and the data available in app stores. We will also seek to develop multi objective predictive models using Search Based Software Engineering (SBSE) [11][13]. The use of multi objective SBSE will allow us to develop predictive models tailored to the conflicting and competing needs of different app store developers and, perhaps also, their customers.

We also believe our data may contain many other interesting relationships between features, prices, ratings and ranks-of-downloads, that have yet to be discovered and reported upon. To facilitate this future work, we make the full dataset available for other researchers to mine, analyse and experiment with. The datasets used in the work reported in this paper can be downloaded from the UCLAppA page:

```
www0.cs.ucl.ac.uk/staff/F.Sarro/
projects/UCLappA/UCLappA.html
```

9 Acknowledgement

The research is funded by Engineering and Physical Sciences Research Council CREST Platform Grant (EP/G060525) and Dynamic Adaptive Automated Software Engineering (DAASE) programme grant (EP/J017515).

References

[1] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In 33^{rd} *International Conference on Software Engineering (ICSE'11)*, pages 1–10, New York, NY, USA, 2011. ACM.

[2] Normi Sham Awang Abu Bakar and Iqram Mahmud. OSS-Grab: Software repositories and app store mining tool. *Lecture Notes on Software Engineering (LNSE) Open Access Journal*, 1(3):219 – 223, November 2013.

[3] David Lavid Ben Lulu and Tsvi Kuflik. Functionality-based Clustering using Short Textual Description: Helping Users to Find Apps Installed on Their Mobile Device. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces (IUI '13)*, pages 297–306, Santa Monica, CA, USA, 19-22 March 2013. ACM.

[4] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-miner: Mining informative reviews for developers from mobile app marketplace. In *36th International Conference on Software Engineering (ICSE 2014)*, pages 767–778, 2014.

[5] Steffen Dienst and Thorsten Berger. Static analysis of app dependencies in android bytecode. Technical report, 2012. technical note.

[6] Horatiu Dumitru, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, and Mehdi Mirakhorli. On-demand Feature Recommendations Derived from Mining Public Product Descriptions. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*, pages 181–190, Hawaii, USA, 21-28 May 2011. ACM.

[7] Ben Eaton, Silvia Elaluf-Calderwood, Carsten Sørensen, and Youngjin Yoo. Dynamic Structures of Control and Generativity in Digital Ecosystem Service Innovation: The Cases of the Apple and Google Mobile App Stores. Working Paper Series 183, The London School of Economics and Political Science, April 2011.

[8] Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. Checking app behavior against app descriptions. In *36th International Conference on Software Engineering (ICSE 2014)*, pages 1025–1035, 2014.

[9] Natalie Gruska, Andrzej Wasylkowski, and Andreas Zeller. Learning from 6,000 Projects: Lightweight Cross-project Anomaly Detection. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '10)*, pages 119–130, Trento, Italy, 12-16 July 2010. ACM.

[10] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering (RE 2014)*, 2014. To appear; available on line.

[11] Mark Harman. The Relationship between Search Based Software Engineering and Predictive Modeling. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering (PROMISE '10)*, pages 1–13, Timisoara, Romania, 12-13 September 2010. ACM.

[12] Mark Harman, Yue Jia, and Yuanyuan Zhang. App Store Mining and Analysis: MSR for App Stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR '12)*, pages 108–111, Zurich, Swiss, June 2012. IEEE.

[13] Mark Harman and Bryan F. Jones. Search Based Software Engineering. *Information and Software Technology*, 43(14):833–839, December 2001.

[14] Ahmed E. Hassan. Mining Software Repositories to Assist Developers and Support Managers. In *Proceedings of the 22nd International Conference on Software Manteinance (ICSM '06)*, pages 339–342, Philadelphia, PA, USA, 24-27 September 2006. IEEE.

[15] Leonard Hoon, Rajesh Vasa, Jean-Guy Schneider, and John Grundy. An analysis of the mobile app review landscape: Trends and implications, 2014. available on line from Swinbourne University of Tethnology, Australia.

[16] http://appworld.blackberry.com/webstore/. BlackBerry App World.

[17] http://officialandroid.blogspot.co.uk/2012/09/google-play-hits-25-billion downloads.html. Android official Blog.

[18] https://play.google.com/store. Google Play.

[19] http://www.apple.com/iphone/apps-for iphone/. The App Store.

[20] http://www.apple.com/pr/library/2013/01/07App-Store-Tops-40-Billion-Downloads-with-Almost-Half-in 2012.html. Apple Press Info - App Store Tops 40 Billion Downloads.

[21] http://www.berryreview.com/2012/05/01/99500-blackberry-apps-now-in-app-world-25-playbook apps/. Berryreview.com.

[22] Claudia Iacob and Rachel Harrison. Retrieving and Analyzing Mobile App Feature Requests from Online Reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*, San Francisco, California, USA, 18-19 May 2013.

[23] Hammad Khalid. On identifying user complaints of iOS apps. In David Notkin, Betty H. C. Cheng, and Klaus Pohl, editors, *35th International Conference on Software Engineering (ICSE 2013)*, pages 1474–1476. IEEE/ACM, 2013.

[24] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed Hassan. What do mobile app users complain about? A study on free iOS apps. *IEEE Software*, 2014. To appear; available online.

[25] M. M. Lehman. On understanding laws, evolution and conservation in the large program life cycle. *Journal of Systems and Software*, 1(3):213–221, 1980.

[26] Soo Ling Lim and Peter J. Bentley. Investigating app store ranking algorithms using a simulation of mobile app ecosystems. In *IEEE Congress on Evolutionary Computation*, pages 2672–2679, 2013.

[27] Mario Linares-Vásquez. Supporting evolution and maintenance of android apps. In *36th International Conference on Software Engineering (ICSE 2014) Doctoral Symposium*, pages 714–717, 2014.

[28] Mario Linares-Vásquez, Andrew Holtzhauer, Carlos Bernal-Cárdenas, and Denys Poshyvanyk. Revisiting android reuse studies in the context of code obfuscation and library usages. In *11th Working Conference on Mining Software Repositories (MSR 2014)*, pages 242–251, 2014.

[29] Edward Loper and Steven Bird. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics (TeachNLP '02)*, pages 69–72, Philadelphia, USA, 7-12 July 2002. Association for Computational Linguistics.

[30] Tim Menzies. Beyond Data Mining; Towards "Idea Engineering". In *Proceedings of the 2nd International NSF sponsored Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE '13)*, San Francisco, USA, 25-26 May 2013.

[31] Roberto Minelli and Michele Lanza. SAMOA - A visual software analytics platform for mobile applications. In *International conference software maintenance (ICSM 2013)*, pages 476–479. IEEE, 2013.

[32] Roberto Minelli and Michele Lanza. Software Analytics for Mobile Applications - Insights & Lessons Learned. In *Proceedings of the 17th European Conference on Software Maintenance and Reengineering (CSMR '13)*, Genova, Italy, 5-8 March 2013. IEEE.

[33] Meiyappan Nagappan, Emad Shihab, and Ahmed E. Hassan. Challenges in mobile apps: A multi-disciplinary perspective. In *Conference of the Center for Advanced Studies on Collaborative Research (CASCON '13)*, pages 378–381, 2013.

[34] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *requirements engineering (RE 2013)*, pages 125–134. IEEE, 2013.

[35] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. WHYPER: Towards Automating Risk Assessment of Mobile Applications. In *Proceedings of the 22nd USENIX Security Symposium*, Washington DC, USA, 14-16 August 2013.

[36] IJ.M. Ruiz, M. Nagappan, B. Adams, and AE. Hassan. Understanding reuse in the android market. In *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, pages 113–122, June 2012.

[37] Israel J. Mojica Ruiz, Meiyappan Nagappan, Adams Bra, Thorsten Berger, Steffam Dienst, and Ahmed E. Hassan. On the relationship between the number of ad libraries in an android app and its rating, 2014. available on line from Queen's University, Canada.

[38] J. Schneidwaind. Big Blue Unveiling. Newspaper, 23 November 1992: 2B.

[39] Weiyi Shang, Bram Adams, and Ahmed E. Hassan. Using Pig as a Data Preparation Language for Large-scale Mining Software Repositories Studies: An Experience Report. *Journal of Systems and Software*, 85(10):2195–2204, October 2012.

[40] Martin J. Shepperd and Steven G. MacDonell. Evaluating prediction systems in software project estimation. *Information & Software Technology*, 54(8):820–827, 2012.

[41] Mark D Syer, Bram Adams, Ying Zou, and Ahmed E Hassan. Exploring the development of micro-apps: A case study on the blackberry and android platforms. In *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on*, pages 55–64. IEEE, 2011.

[42] Mark D. Syer, Meiyappan Nagappan, Bram Adams, and Ahmed E. Hassan. Studying the relationship between source code quality and mobile platform dependence. *Software Quality Journal*, 2014. To appear; available online.

[43] Mark D. Syer, Meiyappan Nagappan, Ahmed E. Hassan, and Bram Adams. Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open-source Android apps. In *Conference of the Center for Advanced Studies on Collaborative Research (CASCON '13)*, pages 283–297, 2013.

[44] Seyyed Ehsan Salamati Taba, Iman Keivanloo, Ying Zou, Joanna Ng, and Tinny Ng. An exploratory study on the relation between user interface complexity and the perceived quality of android applications. In *International Conference on Web Engineering (ICWE 2014))*, 2014. Late Breaking Result.

[45] Joey van Angeren, Vincent Blijleven, Slinger Jansen, and Sjaak Brinkkemper. Complementor embeddedness in platform ecosystems: The case of google apps. In *7th IEEE International Conference on Digital Ecosystems and Technologies (DEST 2013)*, pages 37–42, 2013.

[46] Roy Want. iPhone: Smarter Than the Average Phone. *Pervasive Computing*, 9(3):6–9, July-September 2010.

[47] Toshihiko Yamakami. Foundation-based Mobile Platform Software Engineering: Implications to Convergence to Open Source Software. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09)*, pages 206–211, Seoul, Korea, 24-26 November 2009. ACM.

[48] Yingyuan Yang, Jinyuan Stella Sun, and Michael W. Berry. APPIC: Finding the hidden scene behinddescription files for android apps, 2014. Available on line from University of Tennessee, USA.

[49] Andy Zaidman, Bart Van Rompaey, Serge Demeyer, and Arie van Deursen. Mining Software Repositories to Study Co-Evolution of Production and Test Code. In *Proceedings of the 1st International Conference on Software Testing, Verification, and Validation (ICST '08)*, pages 220–229, Lillehammer, Norway, April 2008. IEEE.