App Store Mining and Analysis: MSR for App Stores

Mark Harman, Yue Jia, and Yuanyuan Zhang
University College London, Malet Place, London, WC1E 6BT, UK.

***Abstract*—This paper introduces app store mining and analysis as a form of software repository mining. Unlike other software repositories traditionally used in MSR work, app stores usually do not provide source code. However, they do provide a wealth of other information in the form of pricing and customer reviews. Therefore, we use data mining to extract feature information, which we then combine with more readily available information to analyse apps' technical, customer and business aspects. We applied our approach to the 32,108 non-zero priced apps available in the Blackberry app store in September 2011. Our results show that there is a strong correlation between customer rating and the rank of app downloads, though perhaps surprisingly, there is no correlation between price and downloads, nor between price and rating. More importantly, we show that these correlation findings carry over to (and are even occasionally enhanced within) the space of data mined app features, providing evidence that our 'App store MSR' approach can be valuable to app developers.**

I. 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 have downloaded them. This provides both qualitative and quantitative data about the customer perception of the apps. Business information is available concerning the downloads and price of apps. Technical information is also available in the descriptions of apps, but it is in free text format, so data mining is required to extract the technical details required for analysis.

This is perhaps a unique situation in the history of software engineering: never before has there been a nexus of readily available information that combines the users' view, the developers' claims and the sales information pertinent to a large corpus of software products from many different providers. The combination of these three types of information provides a rich and inter-related set of data from which we can analyse and understand this new software engineering paradigm of app development. We argue that app store data mining and analysis will support the nascent app development industry, providing insights into the added value of features under consideration for new products and next releases.

To support these claims, we mine and analyse relationships between the technical, business and user perspectives for the Blackberry app store, showing how the findings can be used to inform and guide developers and managers. We study the relationships between three areas of interest: technical (through features offered), customer perceptions (through ratings and download rankings) and business (through price). In order to focus on the relationship between all three of these concerns, we consider only those apps for which there is a non-zero price.

This is the first time that such an empirical analysis of app relationships has been attempted in the literature. With this paper we seek to introduce the study of what might be termed 'App Store Repository Mining', which is closely related to more traditional approaches to Mining Software Repositories, as we explained in the Related Work Section (Section V).

II. App Analysis Framework

Our approach to app store analysis consists of the four phases shown in Figure 1. The first phase extracts raw data from the app store (in this case BLACKBERRY APP WORLD[1], 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 of the available attributes of each app relating to price, ratings and textual descriptions of the app itself. The third phase uses data mining to extract feature information from the textual descriptions and the final phase computes metrics concerning the technical, business and customer information extracted. The rest of this section explains the first three steps of our extraction and analysis approach in more detail.

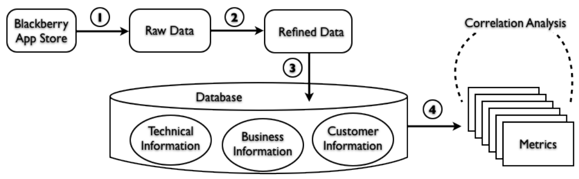

Figure 1. **Overall App Analysis Architecture**: A four phase approach extracts, refines and stores app information for subsequent analysis.

Phase 1 (Data Extraction): We implemented a web crawling system to collect the raw webpage data from the app store. The crawler first collects all category information of the app store and then scans each category page to find the list of addresses of all the apps in each category, using this to locate and extract raw data on each app within each category.

Phase 2 (Parsing): The raw data is parsed according to a set of pattern templates, the attributes of which specify a unique searchable signature for each attribute of interest. Some attribute fields are populated by humans, so we created templates that account for the various ways in which the human might provide the equivalent information. However, once this manual step is complete the entire process is fully automated (until such time that the app store changes structure). We developed patterns to capture information about Category, Description, Price, Customers' Rating, and the Rank of Downloads of each app. To apply our approach to a different app store we need modify only the data extractor and the parsing phase to accommodate the different app store structure and data representations respectively.

Phase 3: (Data Mining Features): There are many ways to define a 'feature'. For our purposes, feature information

[1] http://appworld.blackberry.com/webstore/

is data mined from app descriptions, because we do not have access to app source code. We define a feature to be a property, captured by a set of words in the app description and shared by a set of apps.

Since app descriptions are written in natural language, extracting features from the description text requires data mining techniques more usually associated with Natural Language Processing (NLP). We developed a simple five-step NLP algorithm to extract feature information and implemented it using the Natural Language Toolkit (NLTK), a comprehensive natural language processing package in python [6].

Our feature extraction algorithm is presented as Algorithm 1 below. The first step identifies feature patterns, thereby identifying the 'coarse features' of apps. Fortunately, developers often use informal patterns to list and clarify the features released. A feature pattern consists of three parts: the phrases that signify the start of a feature list, the feature list itself and closing phrase that signifies the end of the feature list.

From the feature list, we filter out noise words, which we determine to be those from the English language STOPWORDS set in the NLTK data package. We then perform a word frequency and co-location analysis to find words that associate frequently, built on top of NLTK's TrigramCollocationFinder classes. This produces a set of 'featurelets'; groups of commonly occurring co-located words. We then cluster featurelets into features using a greedy based clustering algorithm (Algorithm 2 below). The clustering similarity measure is defined in terms of the number of words shared by two featurelets.

Algorithm 1 Feature Extraction Algorithm

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

III. Metrics for App Analysis

In order to compute information about the features of an app, 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 section formalises the definitions of these metrics to support replication and future work[2].

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 $S(f,d) = \{a_1, \ldots, a_m\}$ be the largest 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 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 defined from (app,database) pairs to reals, to a metric F defined from (feature, database) pairs to reals, as follows:

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

Algorithm 2 Greedy Feature Cluster Algorithm

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

IV. Correlation Analysis

We start by exploring three correlations, each for both apps and the features we extract from them. For each we shall use a Spearman's Rank Correlation.

RQ1: What is the correlation between the Price (P) and the Rating (R) for apps and also for the features we extract from them?

RQ2: What is the correlation between the Price (P) and the rank of Downloads (D)?

RQ3: What is the correlation between the Rating (R) and the rank of Downloads (D)?

To answer these first three 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. Summary data concerning the 19 categories in this appstore database and the answers to our three research questions are presented in Table I (leftmost 7 columns). Correlations between Price(P), Rating(R) and Downloads(D) are presented in the rightmost 6 columns for features and for the apps themselves.

Perhaps somewhat surprisingly, we found a correlation between *neither* the price of an app and its rating, *nor* between the price and the downloads of an app. Neither did we find any correlation between price and rating nor between price and downloads for the features we extracted. This finding applies to both the appstore as a whole and to almost all of the categories within it[3]. 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.

However, as can been seen from Table I, we did find a strong correlation between the rating and the downloads of the apps in almost every category (and also within the app store as a

[2]Data from this paper is available at http://www.cs.ucl.ac.uk/staff/Y.Jia/projects/app_store_mining_analysis/.

[3]There is a mild correlation between price and rating for features in the 'Sports and Recreation' Category, but there are not even mild correlations between price and rating nor between price and downloads for any of the other categories.

whole). Even more interestingly, this correlation tends to carry over to (and is sometimes even stronger for) the features we extract using our data mining techniques. This finding may offer useful guidance to developers in determining which features to consider when designing apps. Therefore, we devised a further research question and corresponding experiment to test the true strength of these feature-based correlation findings.
RQ4: What is the chance of producing a similar feature correlation in each category purely at random?

The metric values for a feature are computed as averages over the apps that share the feature. Could it be that our correlations could have been replicated by random sets of apps; 'pseudo features', devoid of any true meaning. If so, then our correlations would be useless.

To answer this question we constructed pseudo features by randomly sampling sets of apps. These pseudo features denote a sample from the population for which the null hypothesis holds (any correlation is merely a randomly occurring artefact of the data). We constructed pseudo feature samples of size 30 for each category and plotted these along with the correlation values for true features and apps in Figure 2.

Using a box plot we can visually assess the degree of significance of the correlations we found. For example, where the correlation found for the true features lies outside of the whiskers of the box plot, this means that the true feature correlation is outside 2.7 standard deviations from the mean for the pseudo features. For a Gaussian distribution this corresponds to rejecting the null hypothesis at the 99% level [7].

In order to ensure that we did not find our correlations simply because they happened to use 'just the right' number of apps per feature, we constructed our pseudo features using the same distribution of numbers of apps as for the true features we extracted. We also repeated the whole experiment with purely random distributions of numbers of apps per pseudo feature. We obtained very similar results for both experiments. However, space only permits us to include one set of box plots[4], so we include those for the 'same size' experiment. In this experiment the pseudo features have an identical size distribution to our true features, so it can only be the composition of true features that yields correlations that are significantly stronger than the pseudo feature sample.

In Table I, correlation between ratings and downloads is at least as strong for features as for apps in 12 out of 19 cases. Furthermore, Figure 2 reveals that more than half of these are highly significant. We can use this analysis to automatically identify those cases where correlation results are most reliable, thereby increasing the actionability of our findings and overall approach. This analysis can point the developer to surprises in app features that they may not have otherwise considered. For example, within the `travel` category, which enjoys a highly significant strong feature correlation, the feature {`near`, `wifi`, `hotspot`}, which one might expect to be important to users, scores lower for both rating and download metrics than the feature {`get`, `nearby`, `restaurants`}. Perhaps travelling users care more about feeding themselves than their devices. This finding might surprise developers.

We can also make observations about the store structure. For example, 'Games', 'Utilities' and 'Sports & Recreation' categories have stronger correlations among apps than features. Our results provide evidence to suggest that these diverse categories may benefit from further refinement to identify more coherent sub-categories; when removed, overall app correlation drops from 0.79 to 0.75, while feature correlation remains unchanged.

[4]The other box plots are available at the website: http://www.cs.ucl.ac.uk/staff/Y.Jia/projects/app_store_mining_analysis/.

V. Related Work

Recent work on mining software repositories has produced scalable techniques for exploring the wealth of source code and associated documentation that can be found in software repositories [5]. This work explores information that can be mined from many sources including emails, change logs, configuration files, user documentation, bug reporting systems and, of course, the source code itself. In this way, a large amount of information is available about the systems under investigation.

If one views an app store as a form repository, then our work can also be thought of as being akin to mining software repositories. However, the technical information we mine is that 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 [1]. Though there has been work on app store analysis [2], we believe that ours is the first paper to data mine and analyse features and their relationship to non-technical information, reformulating the App Store Analysis problem as one for which the MSR community is well placed to exploit.

Previous work on Mining Software Repositories has tended to focus on understanding, predicting and, ultimately guiding and controlling the process of software evolution [4], [9]. Our goal is to extend this, by combining mined technical data with available non-technical user and business data to understand their inter-relationships. The number and granularity of the software products we consider also differs from previous work: Mining Software Repositories typically uses a white box analysis of multiple applications [3] of software products of (sometimes) very large size [8]. By contrast, to mine app stores, we use a black box analysis and are likely to consider potentially many more software products, but of smaller size and without necessarily having available source code.

VI. Conclusion and Future Work

Appspace is very different from traditional software development spaces: the granularity is finer and there is a ready source of information on price, customer rating and, with a little data mining, the features offered by apps. These attributes make appspace ideal for empirical analysis.

Our results demonstrate the value of app store analysis and open up a potentially rich avenue for future Software Repository Mining research. For example, future work can and will surely consider other app stores, data mining algorithms, properties (including non functional properties), metrics and analyses. Our work could also be extended to app store prediction, optimisation and time-series mining and analysis. Future work may also develop models of the evolution of app stores and the apps they contain. We hope that this paper will serve to stimulate this exciting new research agenda of App Store Repository Mining.

Table I
Blackberry App World: The first 7 columns present summary data computed 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 maximum ranks for each category. The final 6 columns present the Spearman rank correlations we computed. We present correlation values for the features we data mined from app descriptions (the three columns labeled 'Feature Correlation') and also the correlations we computed for the apps themselves (the three columns labeled 'App Correlation'). In all 6 of these columns, the single letter labels stand for (P)rice, (R)ating and (D)ownloads.

Name of Categories	Number of	Price (£)	Rank of Downloads			Rating	Feature Correlation			App Correlation		
	Non-free Apps	Mean	Mean	Median	Max	Mean	P,R	P,D	R,D	P,R	P,D	R,D
Reference & eBooks	11,584	4.27	30,388	31,215	1,155	0.12	0.20	0.17	0.76	0.02	0.03	0.83
Themes	10,936	3.12	21,055	21,255	18	1.68	-0.28	-0.45	0.82	-0.10	-0.05	0.83
Games	2,604	2.64	15,919	13,560	153	2.13	-0.31	0.06	0.47	-0.17	-0.21	0.81
Utilities	1,362	4.61	16,294	13,998	63	2.32	0.19	0.31	0.40	0.33	0.43	0.81
Entertainment	908	5.76	18,413	16,376	134	1.86	0.25	0.39	0.83	-0.10	-0.01	0.76
Travel	764	4.81	25,439	26,113	553	0.67	-0.12	-0.11	0.88	-0.28	-0.26	0.85
Health & Wellness	626	15.95	19,852	18,296	266	1.58	-0.52	-0.42	0.61	-0.21	0.02	0.63
Education	576	5.68	22,222	21,768	1,595	1.38	0.20	0.06	0.87	-0.06	0.01	0.78
Productivity	503	6.32	15,124	11,924	252	2.54	-0.31	-0.28	0.76	0.42	0.33	0.76
Music & Audio	499	2.05	24,523	27,248	204	0.99	-0.25	-0.32	0.82	0.07	0.16	0.79
Photo & Video	393	2.51	21,126	22,879	15	1.40	-0.33	-0.25	0.91	0.02	0.06	0.82
Business	350	12.57	19,063	18,032	817	1.79	0.03	0.08	0.88	0.01	0.08	0.73
Maps & Navigation	245	12.90	17,140	13,909	655	2.16	0.06	-0.29	0.77	0.09	0.13	0.32
Sports & Recreation	239	4.81	18,808	16,019	943	2.05	0.60	-0.36	0.26	0.26	0.21	0.67
Finance	193	4.38	19,593	16,619	251	1.93	0.01	0.37	0.76	-0.10	-0.02	0.77
IM & Social Networking	150	4.42	14,242	11,628	22	2.55	0.15	0.02	0.90	0.16	0.15	0.81
News	73	2.40	17,485	15,391	1,393	1.73	0.38	0.29	0.95	0.04	-0.02	0.75
Weather	58	7.51	12,392	10,642	309	2.44	-0.06	-0.07	0.92	-0.10	-0.03	0.77
Shopping	45	2.70	14,785	11,708	2,543	2.33	-0.38	-0.26	0.38	0.07	0.12	0.54
All Categories	32,108	4.21	23,651	24,329	15	1.17	0.07	-0.09	0.89	0.10	0.12	0.79

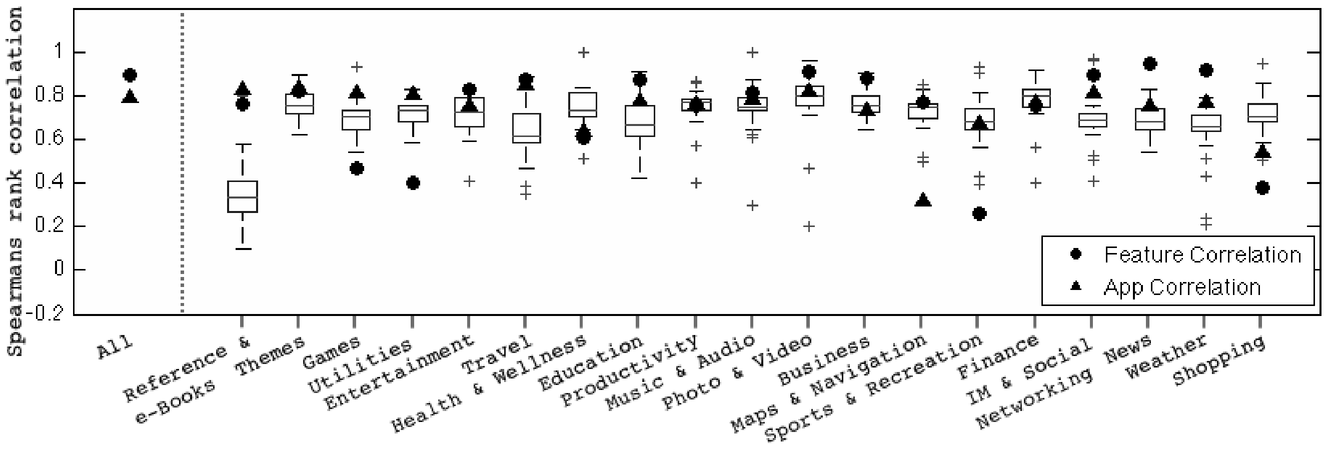

Figure 2. **Significance of Spearman Rank Correlations**. The box plots show the distributions of correlations between customer rating and downloads obtained from a sample of 30 randomly generated 'pseudo features'. This can be visually compared to the true feature correlation values (solid circles) and the true app correlation values (solid triangles). Where true correlation values lie outside of the box, the correlation can be thought of as significant, whereas those that lie outside of the whiskers are very highly significant (equating to approximately the 99% confidence interval).

REFERENCES

[1] H. Dumitru, M. Gibiec, et al. On-demand Feature Recommendations Derived from Mining Public Product Descriptions. In *ICSE '11*, pp. 181–190. ACM, Hawaii, USA, 21-28 May 2011.

[2] B. Eaton, S. Elaluf-Calderwood, et al. 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.

[3] N. Gruska, A. Wasylkowski, et al. Learning from 6,000 Projects: Lightweight Cross-project Anomaly Detection. In P. Tonella and A. Orso, eds., *ISSTA '10*, pp. 119–130. ACM, Trento, Italy, 12-16 July 2010. ISBN 978-1-60558-823-0.

[4] A. E. Hassan. Mining Software Repositories to Assist Developers and Support Managers. In *ICSM '06*, pp. 339–342. IEEE, Philadelphia, PA, USA, 24-27 Sept. 2006.

[5] —. The Road Ahead for Mining Software Repositories. In *FoSM '08*, pp. 48–57. IEEE, Beijing, China, Sept. 28-Oct. 4 2008.

[6] E. Loper and S. Bird. NLTK: The Natural Language Toolkit. In *TeachNLP '02*, pp. 69–72. Association for Computational Linguistics, 7-12 July 2002.

[7] L. D. Mueller and L. Altenberg. Statistical Inference on Measures of Niche Overlap. *Ecology*, 66(4):1204–1210, 1985.

[8] W. Shang, B. Adams, et al. Using Pig as a Data Preparation Language for Large-Scale Mining Software Repositories Studies: An Experience Report. *JSS*, 2011. To appear.

[9] A. Zaidman, B. V. Rompaey, et al. Mining Software Repositories to Study Co-Evolution of Production and Test Code. In *ICST '08*, pp. 220–229. IEEE, Lillehammer, Norway, 9-11 April 2008.