UCL DEPARTMENT OF
COMPUTER SCIENCE

Research Note
RN/13/21

Mining App Stores: Extracting Technical, Business and Customer Rating Information for Analysis and Prediction

November 5, 2013

Anthony Finkelstein

Mark Harman

Yue Jia

Federica Sarro

Yuanyuan Zhang

Abstract

This paper formulates app store analysis as an instance of software repository mining. We use data mining to extract feature information, together with more readily available price and popularity information, to support analysis that combines technical, business and customer facing app store properties. We applied our approach to 32,108 non-zero priced apps available from the Blackberry app store. 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. We provide empirical evidence that our extracted features are meaningful and valuable: they maintain correlations observed at the app level and provide the input to price prediction system that we construct using Case Based Reasoning. Our prediction system statistically significantly outperforms recommended existing approaches to price estimation (and with at least medium effect size) in 16 out of 17 of Blackberry App Store categories.

Mining App Stores: Extracting Technical, Business and Customer Rating Information for Analysis and Prediction

Anthony Finkelstein, Mark Harman, Yue Jia, 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 required for analysis.

This is perhaps a unique situation in software engineering research: never before has there been a collection 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 the 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.

The primary contributions of this paper are:

1) This is the first approach to mining App Stores[1];
2) We empirically investigate the correlations between price, rating and popularity;
3) We empirically investigate price, rating and popularity correlations extended to the mined features';
4) We present an empirical validation of the mined features as the basis of useful price prediction systems, the primary findings of which are:
   a) estimations of price based on mined features and Case Based Reasoning (CBR) outperform both the random guess baseline and current industry best price prediction practice;
   b) a few analogies only are sufficient to CBR to obtain accurate estimations of App price.

The rest of the paper is organised as follows: Section 2 introduces the overall app analysis architecture. Section 3 proposes metrics that capture the attributes of a feature. Section 4 briefly describes Case Based Reasoning, the technique we employed to estimate app prices. Section 5 describes the design of our empirical study, the results of which are presented in Section 6. Section 7 analyses the limitations of the present study, while Section 8 describes other work related to ours. Section 9 describes the actionable findings and future work from our study for the app markets and for the research community respectively. Section 10 concludes.

• *A. Finkelstein, M. Harman, Y. Jia, 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, f.sarro, yuanyuan.zhang}@ucl.ac.uk

1. Strictly speaking the *first* publication on App Store Mining was the MSR 2012 paper [1].This is a considerably extended version of that 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 of 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 the first three steps of our approach in more detail, while Section 3 presents the metrics we introduced to analyse the mined information.

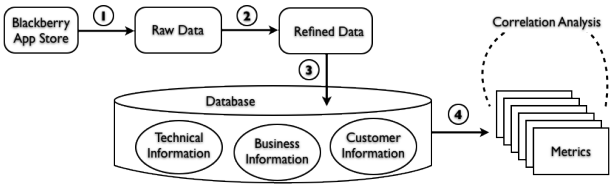

Fig. 1. **Overall App Analysis Architecture**: A four phase approach extracts, refines and stores app information for subsequent analysis.

Phase 1 (Data Extraction): We implemented a web crawling system to collect the raw webpage data from the app store. The crawler first collects all category information from 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. 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 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 [2].

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 correlation analysis and to obtain price estimates, based on features. The mined information could, of course, support many other analyses, and we make all our data available to support such other downstream analyses (See Section 9).

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.

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

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
```

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[3].

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 set $\{a_1, \ldots, a_m\}$ 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 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 reals, to a metric F defined from (feature, database) pairs to reals, as follows:

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

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

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

to one of type

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

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 (featureClusters) - 1 do
    currCluster = greedyClusters[i]
    for j = 0 → 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
```

4 Case Based Reasoning

To assess whether mined features can be useful as basis for price estimation we developed a Case Based Reasoning (CBR) system based on these features. In this section we describe CBR in overview. In Section 5.3 we explain the choice we made.

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

CBR is a branch of Artificial Intelligence where knowledge of similar past cases is used to solve new cases [24]. In our work cases represent the apps, which are characterised by a set of n features and stored in a case base.

Given a new app (i.e., target case) — characterised in terms of the same n features — the past apps relevant to solving it are retrieved from the case base. In particular, the relevant cases are identified by employing a similarity function that measures the distance between the target case and the other cases based on the values for the n features of these applications. Although numerous techniques are available to measure similarity, unweighted Euclidean distance has been the most widely used in software engineering.

Usually a subset of features can be identified to determine a suitable subset that give the most accurate estimation. Some existing tools, e.g. ANGEL [3], offer this functionality by applying a brute force algorithm (i.e., searching for all possible feature subsets), however as reported in [3][4] this approach is too computationally expensive for large datasets.

The number of cases (i.e., the number of most similar applications) to be used to generate the estimation is determined by the user. Earlier studies in software engineering restricted their analysis to the closest case (k=1) (see e.g., [4][6]). More recent approaches have employed different numbers of analogies to search for the best value to select for this parameter to the overall CBR approach [7][8].

Once the k most relevant cases have been retrieved, an adaptation strategy is usually employed to obtain a prediction for the target case. Many adaptation techniques can be used, such as the nearest neighbour, the mean, the inverse distance weighted mean and inverse rank weighted mean of the closest cases [6].

5 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.

5.1 Research Questions

The first three research questions investigate the correlation between price, rating and popularity (rank of downloads) for apps and for the features we mined from the app descriptions.

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

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

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

The metric values for a feature are computed as averages over the apps that share the feature. This raises the question as whether our correlations could have been replicated by random sets of apps, 'pseudo features', devoid of any true meaning. Thus, in this case our correlations would be useless. In order to test whether the correlations for the mined features are robust and reliable, we seek an answer to the following research question.

RQ4: Feature Meaningfulness. How meaningful are the features extracted?

While the features we extract tend to have conceptually appealing names, such as {`near`, `wifi`, `hotspot`}, we need to know whether they can be relied upon as technical items that carry meaning. We address this in two related ways: we ask whether the statistical correlations observed at the app level carry over to the features we extract. If corrections are maintained, then there is at least some evidence that the features may be capturing intrinsic properties of the apps. We also ask what the chance is that such correlations could be observed by features constructed entirely at random, rather than through data mining. This allows us to give statistical confidence intervals to the feature constriction process. These two questions are RQs 4.1 and 4.2 below:

RQ4.1: Correlation Carry Over. Do correlations observed at the app level carry over to the feature level?

RQ4.2: Significance of Feature Construction. What is the chance of producing a similar feature correlation in each category purely at random?

The last four research questions (i.e., RQs 5-8) concern the use of the mined features to predict app price based on reasoning by analysis (Case Based Reasoning). If the predictive quality is high, then the obtained estimations may be useful in themselves (to help guide pricing decisions, for example). Such estimations would also validate the quality of the feature information mined (because they would be extracted to provide accurate predictions of important attributes such as price).

RQ5: Baseline Comparison. Can our learned prediction system outperform Random Guessing?

Random guessing is a naïve benchmark suggested to assess the usefulness of a prediction system [9]. It randomly assigns the y value of another case to the target case. More formally, it is defined as: predict a y for the target case t by randomly sampling (with equal probability) over all the remaining $n-1$ cases and take $y = r$ where r is drawn randomly from $1...n^r = t$ [9]. Any prediction system should outperform random guessing since an inability to predict better than random implies that the prediction system is not using any target case information [9].

RQ6: State of the Art Comparison. How well do our prediction system perform against current market price approaches?

This question aims to compare the effectiveness of using our price recommender prediction system with respect to the current market price approaches. A quick glance at the 'Top Paid apps list' of the app store shows that, at the time of writing, three of the top 25 apps are currently selling for the minimum price of $0.99 so a developer could be tempted to sell apps at the minimum price [10] (in the following we refer to this approach as MinPrice). Another common pricing practice [11] [12] uses the mean or the median of the prices of apps that belong to a same category [13]. We refer to this approaches as MeanPrice and MedianPrice. Thus, the research questions we would like to answer are the following:

- RQ6.1: Does CBR outperform MinPrice?
- RQ6.2: Does CBR outperform MeanPrice?
- RQ6.3: Does CBR outperform MedianPrice?

RQ7: Competitiveness of Current Practices. How well do current market price approaches compare to Random Guessing?

We ask this question in order to assess the effectiveness of the state of the art. In particular, the research questions we would like to answer are the following:

- RQ7.1: Does MinPrice outperform Random Guessing?
- RQ7.2: Does MeanPrice outperform Random Guessing?
- RQ7.3: Does MedianPrice outperform Random Guessing?

RQ8: CBR Parameter Tuning. How sensitive is our approach to the choice of the k most relevant cases?

We ask this question in order to assess how sensitive our approach is to the number of analogies used. In particular, we compared the accuracy achieved by CBR exploiting different numbers of most relevant cases (i.e., from 1 up to 15 analogies) to obtain the final price prediction.

5.2 Data Employed in the Empirical Study

To answer the first four 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. Summary data concerning the 19 categories in this appstore database and the answers to our first three research questions are presented in Table 1 (leftmost 8 columns). Correlations between Price(P), Rating(R) and Downloads(D) are presented in the rightmost 6 columns for features and for the apps themselves.

The last four research questions concern 17 out of 19 categories in this app store database. The 'Reference & eBooks' category and the 'Themes' category are excluded from our analysis, because these categories contain pseudo apps which differ primarily on content and not on features. Summary data concerning the employed categories are presented in Table 1.

5.3 Evaluation Criteria and Validation Method

To answer RQs 1-3 (i.e., investigate the correlation between price, rating and popularity of apps and features) we employed a Spearman's Rank Correlation Test. This statistic ranges from -1 to +1, where -1 indicates a perfect inverse correlation and +1 indicates perfect correlation. No correlation is indicated by 0.

To answer RQ4.1 we use Spearman's Rank Correlation Tests at the feature level and compare these to those observed for the app level. To answer RQ4.2 we constructed pseudo features by randomly sampling sets of apps as detailed in section 6.2. Then we used box plots to visually assess the degree of significance of the correlations we found and compare them with respect to the correlation values obtained for the true features.

To answer RQs 5-8 (i.e., assess the accuracy of the price estimations obtained by using CBR and the mined features) we followed the framework recently proposed by Shepperd and MacDonell [9] for evaluating competing prediction systems. This framework suggests three fundamental questions to assess a prediction system. In particular, for a given accuracy statistic S and candidate prediction systems P_1 and P_2 one must ask:

1) Does the prediction system P_i outperform a baseline of random guessing? If the answer is not yes then it cannot even be claimed that P_i is predicting at all since it performs worse than random [9].

2) Is the difference among P_1 and P_2 statistically significant in term of S? In other words how likely is any observed effect to have occurred by chance [9]?
3) Is the effect size large enough to justify the use of a prediction system in practice? It may be that any improvement that P_2 offers with respect to P_1 is so low as to not be worth the effort to use it [9].

To answer the first question Shepperd and MacDonell [9] suggest measuring accuracy as the Mean of Absolute Residuals (MAR) relative to random guessing P_0. They advocate doing this with a standardised accuracy measure (denoted as SA) which, for a prediction technique P_i, compares the results with random guessing as follows:

$$SA_{P_i} = (1 - \frac{MAR_{P_i}}{MAR_{P_0}}) * 100 \quad (1)$$

where MAR_{P_0} is the mean value of a large number, typically 1000, runs of random guessing (denoted as RG in the following). Thus, SA represents how much better P_i is than random guessing and a value close to zero means that the prediction system P_i is practically useless, performing as well as a random guess.

To check for statistical significance we used the Wilcoxon test [14] since the Shapiro test [15] showed that many of our samples came from non-normally distributed populations, making the T-test unsuitable. Using the Wilcoxon test is a safe test to apply (even for normally distributed data), since it raises the bar for significance, by making no assumptions about underlying data distributions. In particular, we tested the following null hypothesis:

> "*The absolute residuals provided by the prediction system P_i significantly less that those provided by the prediction system P_j.*"

and set the confidence limit, α, at 0.05 and applied the standard Bonferroni correction (α/K, where K is the number of hypothesis) in cases where multiple hypotheses were tested.

To assess whether the effect size is worthy of interest we employed a non-parametric effect size measure, namely the Vargha and Delaney's A_{12} statistic [16], since not all samples were normally distributed. Indeed, as suggested in [9][17], it is better in cases such ours to use a standardised measure rather than a pooled measure such as the Cohen's d effect size. Given a performance measure M, the A_{12} statistic measures the probability that running algorithm A yields better M values than running another algorithm B based on the following formula

$$A_{12} = (R_1/m - (m+1)/2)/n \quad (2)$$

where R_1 is the rank sum of the first data group we are comparing and m and n are the number of observations in the first and second data sample, respectively. If the two algorithms are equivalent, then A_{12} = 0.5. According to Vargha and Delaney [16] a small, medium, and a large difference between two populations is indicated by A_{12} over 0.56, 0.64, and 0.71, respectively.

To validate the price estimations obtained with the prediction system considered, we used a 10-fold cross-validation that partitions the initial dataset of n observations in 10 randomly selected test subsets of equal size. For each test subset, we use the remaining observations as the training set which is exploited to build the estimation model that is then validate on the corresponding test subset.

5.4 Case Based Reasoning Setting

To apply CBR we have to choose the attribute set describing the applications, the similarity function, the number of analogies to consider and the analogy adaptation strategy for obtaining the final estimation.

In this study, we used the features mined from the app store for each category as the set of attributes to use in determining the values that pertain to a 'case'. That is, we form a vector of values which consists of a bit string that contains one bit for each feature that an app may possess: 1 if the app possesses the feature; 0 if not. As an example, if an app resides in a category with 40 potential features (such as the case of all games apps), then the vector has 40 bits denoting the presence or absence of these 40 features.

We used ANGEL [3] to implement our CBR. ANGEL supports the Euclidean distance measure between vectors and we used this metric to compute apps similarity, while the final price estimation has been computed as the mean price of the k nearest apps (i.e., analogies). We report results of each of the choices of k, between $k = 1$ and $k = 15$ analogies. To answer RQ5, RQ6 and RQ7 we consider the worst, the mean and the best results (given the choice of k), while in Section 6.6 we report an analysis of the performance of CBR for each k ($1 \leq k \leq 15$) to answer to RQ8. We did not used the feature subsect selection algorithm provided by ANGEL since it is too computationally expensive for large datasets as the ones employed in this work.

TABLE 1

Blackberry App World: The first 8 columns present summary data computed for each category. The number of non-free apps and the mean price 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 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 Non-free Apps	Number of Features	Price (£) Mean	Rank of Downloads			Rating	Feature Correlation			App Correlation		
				Mean	Median	Min	Mean	P,R	P,D	R,D	P,R	P,D	R,D
Reference & eBooks	11,584	82	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	33	3.12	21,055	21,255	18	1.68	-0.28	-0.45	0.82	-0.10	-0.05	0.83
Games	2,604	40	2.64	15,919	13,560	153	2.13	-0.31	0.06	0.47	-0.17	-0.21	0.81
Utilities	1,362	78	4.61	16,294	13,998	63	2.32	0.19	0.31	0.40	0.33	0.43	0.81
Entertainment	908	86	5.76	18,413	16,376	134	1.86	0.25	0.39	0.83	-0.10	-0.01	0.76
Travel	764	81	4.81	25,439	26,113	553	0.67	-0.12	-0.11	0.88	-0.28	-0.26	0.85
Health & Wellness	626	84	15.95	19,852	18,296	266	1.58	-0.52	-0.42	0.61	-0.21	0.02	0.63
Education	576	82	5.68	22,222	21,768	1,595	1.38	0.20	0.06	0.87	-0.06	0.01	0.78
Productivity	503	92	6.32	15,124	11,924	252	2.54	-0.31	-0.28	0.76	0.42	0.33	0.76
Music & Audio	499	82	2.05	24,523	27,248	204	0.99	-0.25	-0.32	0.82	0.07	0.16	0.79
Photo & Video	393	86	2.51	21,126	22,879	15	1.40	-0.33	-0.25	0.91	0.02	0.06	0.82
Business	350	96	12.57	19,063	18,032	817	1.79	0.03	0.08	0.88	0.01	0.08	0.73
Maps & Navigation	245	71	12.90	17,140	13,909	655	2.16	0.06	-0.29	0.77	0.09	0.13	0.32
Sports & Recreation	239	239	4.81	18,808	16,019	943	2.05	0.60	-0.36	0.26	0.26	0.21	0.67
Finance	193	75	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	78	4.42	14,242	11,628	22	2.55	0.15	0.02	0.90	0.16	0.15	0.81
News	73	53	2.40	17,485	15,391	1,393	1.73	0.38	0.29	0.95	0.04	-0.02	0.75
Weather	58	73	7.51	12,392	10,642	309	2.44	-0.06	-0.07	0.92	-0.10	-0.03	0.77
Shopping	45	56	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	1256	4.21	23,651	24,329	15	1.17	0.07	-0.09	0.89	0.10	0.12	0.79

6 Result Analysis

6.1 RQ1-3. Three correlations for apps

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. This finding applies to both the appstore as a whole and to almost all of the categories within it[4]. 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.

As can been seen from Table 1, 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 whole).

Thus, in answer to RQ1-3: our results show that there is a strong correlation between customer rating and the rank of app downloads and there is no correlation between price and downloads, nor between price and rating.

6.2 RQ4. Feature Meaningfulness

Encouragingly, correlations observed for apps *do* tend to carry over to (and are sometimes even stronger for) the features we extract using our data mining. The Answer to RQ4.1 is thus 'yes'. As can been seen from Table 1, we find 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). In general, our results show that there is a strong 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.

To answer RQ4.2, 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

4. 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.

occurring artefact of the data). In particular, 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 [18].

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[5], 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 1, 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; a finding that might surprise developers more inclined to favour bandwidth over food! This observation is intended merely as one (hopefully slightly amusing) illustration of the kinds of insight that can be uncovered using App Store Mining.

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 subcategories; when removed, overall app correlation drops from 0.79 to 0.75, while feature correlation remains unchanged.

Thus, in answer to RQ4: we have empirical evidence that mined features are meaningful, because our results show that app correlation findings (RQ3) carry over to (and are even occasionally enhanced within) the space of data mined app features. We also find that these correlations are generally much statistically significantly stronger than those that would be observed for randomly constricted features.

So far, we have presented evidence that the mined features are meaningful and that they are potentially useful to developers as sources of insights into customers' implicit and explicit feature requirements, expressed through downloads and ratings respectively. These findings indicate that App Store Mining can be used as one bridge between technical aspects (features) and customer requirements. Our remaining research questions investigate whether our mined features may also be useful as the source of price prediction systems, thereby also going some way towards bridging the gap between development activities and business concerns.

6.3 RQ5. Baseline Comparison

Figure 3 reports the values of Mean Absolute Residuals (MAR) achieved by using CBR and 1000 Random Guessing trials (RG). In particular, we report, for each category, the MAR obtained by employing CBR with the best and worst number of analogies (denoted as CBR (worst k) and CBR (best k)) and also the mean of the MAR obtained by employing CBR with 1 up to 15 analogies (denoted as CBR (mean k)).

We can observe that, in all cases, the MAR values provided by CBR are much lower than those provided by Random Guessing, thus indicating a better prediction accuracy. When we consider the standardised accuracy (SA) values reported in Table 2, we can see that depending on the considered category CBR allowed us to obtain an improvement with respect to Random Guessing ranging from 52% to 61% when CBR (worst k) is considered, from 7% to 96% in case of CBR (mean k) and from 68% to 96% in case of CBR (best k). Thus, even the worst of our price estimates are almost half as good again as can be obtained by the baseline.

These results are confirmed by the Wilcoxon Test performed on the absolute residuals provided by CBR and Random Guessing. Indeed, as we can observe from the results reported in Table 2, CBR provided significantly less absolute residuals than RG with high effect size for almost all the cases. The above results address the baseline comparison question.

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

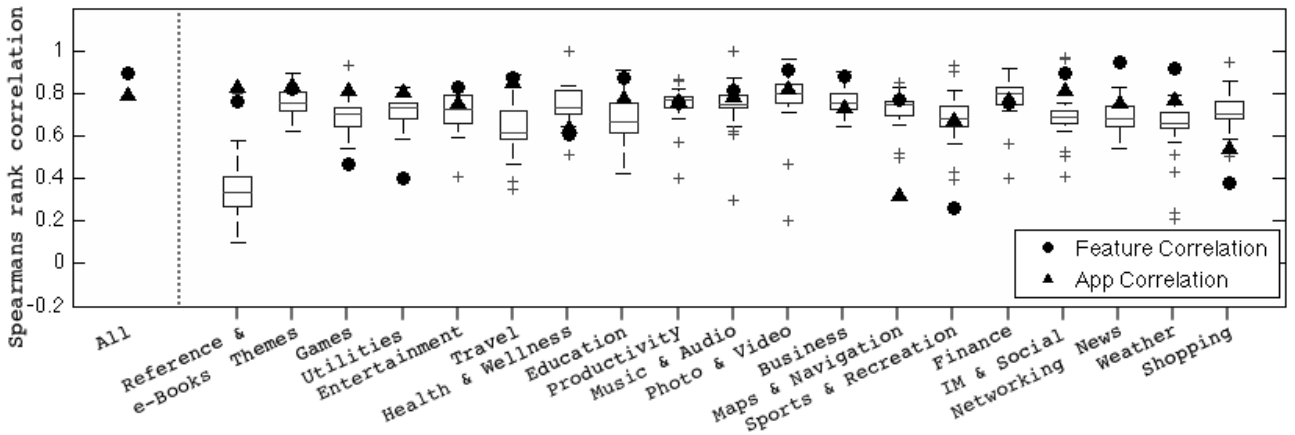

Fig. 2. **Significance of Spearman Rank Correlations**. The box plots show the distributions of correlations between customer rating and downloads obtained from a sample of 30 randomly generated 'pseudo features'. This can be visually compared to the true feature correlation values (solid circles) and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

TABLE 2
Comparison of CBR (worst, mean and best k) vs. Random Guessing (RG) using Standardised Accuracy (SA) and Wilcoxon test (effect size in parentheses).

Category	CBR (worst k) vs. RG		CBR (mean k) vs. RG		CBR (best k) vs. RG	
	SA	Wilcoxon Test	SA	Wilcoxon Test	SA	Wilcoxon Test
Games	85	<0.001 (0.937)	89	<0.001 (0.974)	90	<0.001 (0.975)
Utilities	79	<0.001 (0.940)	82	<0.001 (0.952)	84	<0.001 (0.955)
Entertainment	96	<0.001 (0.990)	96	<0.001 (0.990)	96	<0.001 (0.990)
Travel	93	<0.001 (0.990)	95	<0.001 (0.990)	96	<0.001 (0.990)
Health & Wellness	89	<0.001 (0.957)	90	<0.001 (0.962)	91	<0.001 (0.966)
Education	93	<0.001 (0.990)	94	<0.001 (0.990)	95	<0.001 (0.990)
Productivity	70	<0.001 (0.897)	77	<0.001 (0.925)	79	<0.001 (0.934)
Music & Audio	84	<0.001 (0.947)	88	<0.001 (0.956)	89	<0.001 (0.965)
Photo & Video	80	<0.001 (0.950)	89	<0.001 (0.976)	91	<0.001 (0.978)
Business	64	<0.001 (0.932)	65	<0.001 (0.927)	72	<0.001 (0.927)
Maps & Navigation	82	<0.001 (0.951)	84	<0.001 (0.952)	85	<0.001 (0.957)
Sports & Recreation	86	<0.001 (0.955)	89	<0.001 (0.990)	91	<0.001 (0.967)
Finance	70	<0.001 (0.939)	72	<0.001 (0.990)	74	<0.001 (0.948)
IM & Social Networking	73	<0.001 (0.908)	77	<0.001 (0.921)	78	<0.001 (0.927)
News	62	<0.001 (0.771)	67	<0.001 (0.788)	73	<0.001 (0.808)
Weather	52	<0.001 (0.823)	61	<0.001 (0.884)	68	<0.001 (0.890)
Shopping	58	<0.001 (0.893)	66	<0.001 (0.920)	74	<0.001 (0.904)

Thus, in answer to RQ5: our learned prediction system significantly outperforms baseline Random Guessing for all categories with high effect size.

6.4 RQ6. State of the Art Comparison

Figure 4 reports the Mean of Absolute Residuals (MAR) obtained by CBR and current market price strategies (i.e., MinPrice, MeanPrice, MedianPrice) on all categories. We observe that CBR (worst, mean and best k) achieved the lowest MAR values on all the categories we used, indicating that it provided more accurate prediction with respect to MinPrice, MeanPrice and MedianPrice strategies. Among the market price strategies, we observe that MedianPrice provided better MAR than MinPrice and MeanPrice on 15 out 17 categories, while there is no clear winner between MinPrice and MeanPrice.

Table 3 reports the p-values obtained with the Wilcoxon Test to verify whether the absolute residuals achieved by CBR (with the worst, the mean and the best number of analogies) were significantly less than those provided by the market price strategies. We observe that the results confirm those achieved in terms of MAR, highlighting that CBR achieved absolute residuals significant better than those provided by these approaches in 147 out 153 cases with medium and large effect size, while no statistically significant difference

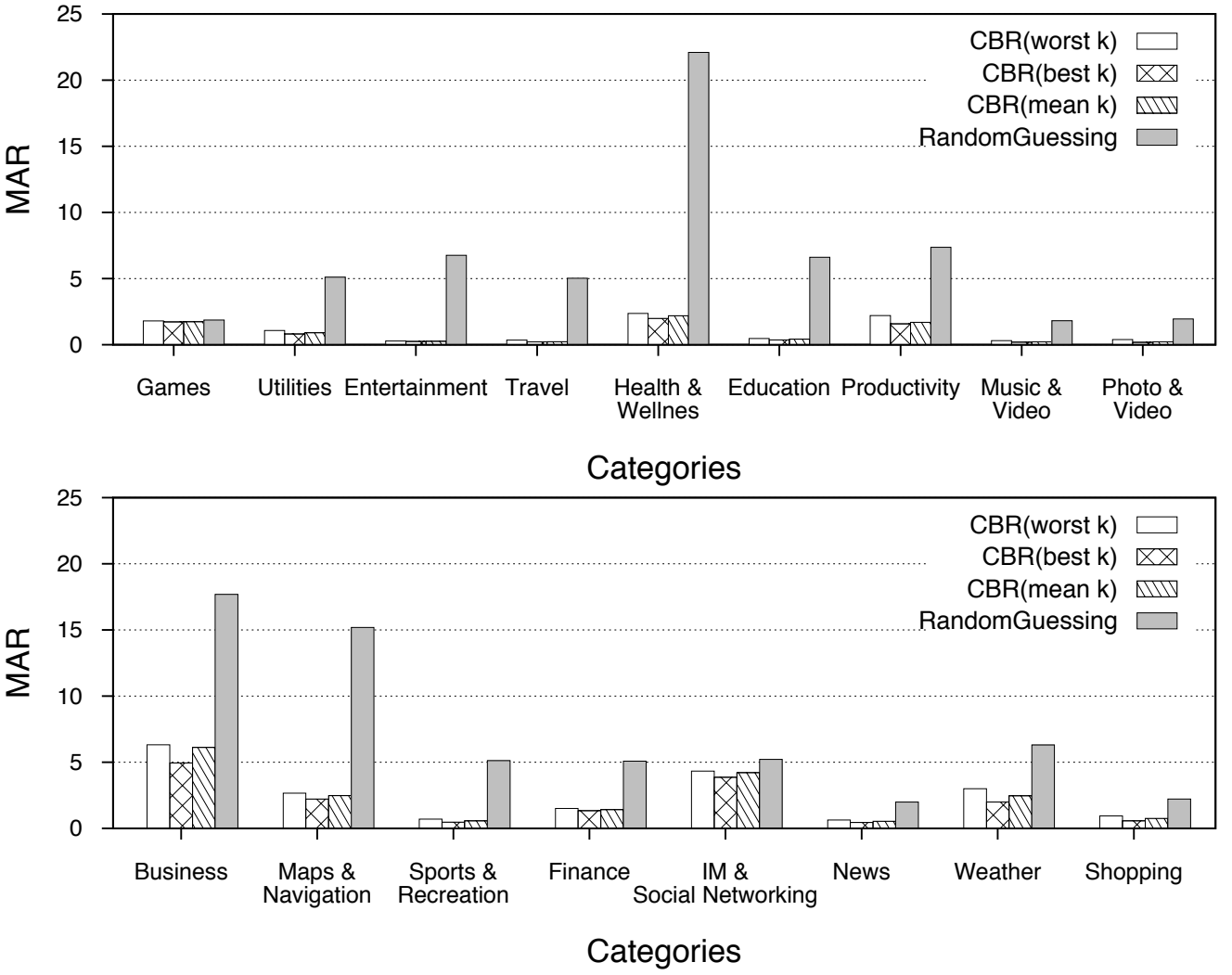

Fig. 3. Comparing the MAR values provided by CBR (best, worst and mean results) and Random Guessing over each of the 17 categories of the Blackberry App Store.

was found in the remaining 6 cases (i.e., vs. MeanPrice and MedianPrice on the category `Music & Video`). In particular, CBR outperformed MinPrice in 48 out 51 cases.

Turning to the comparison with MeanPrice we observe that CBR provided significantly better absolute residuals in all 51 cases always with high effect sizes. Finally, CBR outperformed MedianPrice in 48 out 51 cases with small and medium effect sizes when CBR (worst k) is employed and medium and high effect sizes when CBR (best k) and CBR (mean k) are employed.

Thus, in answer to RQ6: our approach significantly outperforms current market price strategies considered, often with high effect size.

6.5 RQ7. Competitiveness of Current Practices

Table 4 reports the values of standardised accuracy (SA) obtained by the market price strategies considered (i.e., MinPrice, MeanPrice and MedianPrice) with respect to the results obtained with Random Guessing (RG). We can observe that the three strategies perform better than Random Guessing except for MinPrice on the category `Weather` which provides negative results (indicating that, in this case, MinPrice is worse than RG).

On the other hand, MedianPrice provided the highest SA values over all categories. In particular, the average improvement over RG achieved over all categories was 26% for MinPrice, 23% for MeanPrice and 34% for MedianPrice. If we compare these values with those obtained by CBR (see Table 2) we observe that the average improvement with respect to RG provided by CBR over all categories (i.e., 69% with the worst k, 73% with the mean k, and 76% with the best k) is about two times the improvement provided by the MedianPrice and approximately three times the improvement provided by both MinPrice and MeanPrice.

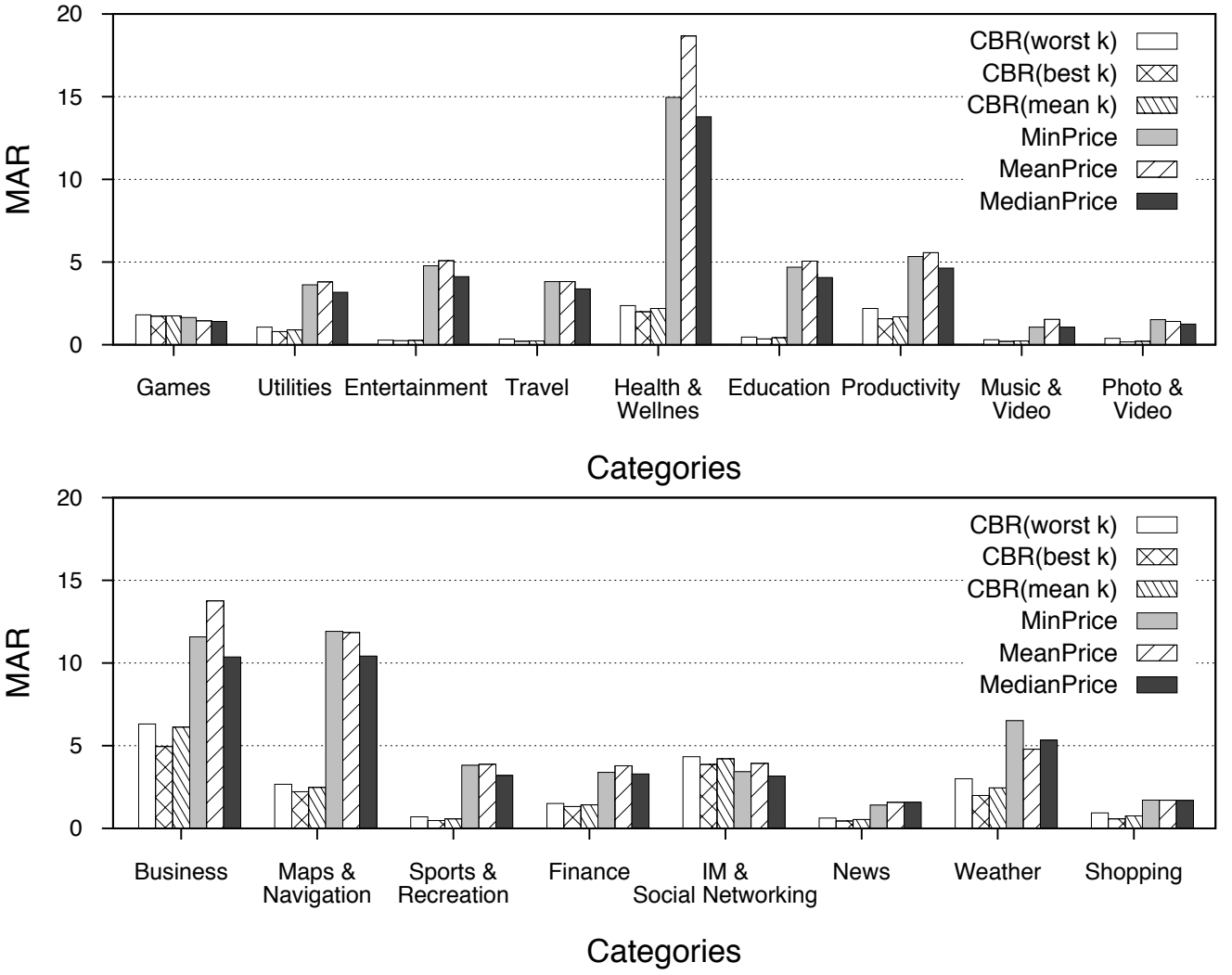

Fig. 4. Comparing the MAR values provided by CBR (best, worst and mean results) and current market price strategies over each of the 17 categories of the Blackberry App Store.

These results are confirmed by the Wilcoxon Test performed on the absolute residuals provided by the considered market pricing strategies and RG. Indeed, as we can observe from Table 4, all these approaches achieved absolute residuals significantly better (i.e., less) than those achieved by RG except for MinPrice on the `Weather` category where no significant difference was found. The effect sizes were small, medium and large for MinPrice and MeanPrice, while they were medium and large for MedianPrice. By contrast the effect sizes achieved by CBR (see Table 2) were always large (i.e., ranging from 0.745 to 0.990) except for the category `Games`.

Thus, in answer to RQ7: the market price strategies we considered (i.e., MinPrice, MeanPrice and MedianPrice) outperform the baseline of Random Guessing.

6.6 RQ8. CBR Parameter Tuning

Table 5 reports the results in terms of Mean Absolute Residuals (MAR) obtained with CBR employing 1 up to 15 analogies (denoted as CBRk, where k is the number of the employed analogies). This allows us to investigate how sensitive our approach is to this parameter (i.e., k) and also to determine a suitable recommendation for the value to be used by app store developers. In general, we can observe a low variation in the MAR values obtained by using different number of analogies as confirmed by the low standard deviation values reported in Table 5.

We applied the Wilcoxon Test to assess whether there was significant difference among the results provided by using CBR with different numbers of analogies. In particular, we verified the following hypothesis: "The absolute residuals provided by CBRk are significantly better (less) than those provided by CBR$k+1$" for $k = 1...14$.

TABLE 3
Comparison of the absolute residuals provided by CBR (worst, mean and best k) and the market price strategies considered (i.e., MinPrice, MeanPrice and MedianPrice) using the Wilcoxon test (effect size in parentheses).

Category	CBR (worst k) vs. MinPrice	CBR (mean k) vs. MinPrice	CBR (best k) vs. MinPrice
Games	<0.001 (0.707)	<0.001 (0.707)	<0.001 (0.707)
Utilities	<0.001 (0.707)	<0.001 (0.699)	<0.001 (0.714)
Entertainment	<0.001 (0.787)	<0.001 (0.787)	<0.001 (0.789)
Travel	<0.001 (0.747)	<0.001 (0.749)	<0.001 (0.752)
Health & Wellness	<0.001 (0.820)	<0.001 (0.814)	<0.001 (0.830)
Education	<0.001 (0.771)	<0.001 (0.772)	<0.001 (0.777)
Productivity	<0.001 (0.719)	<0.001 (0.729)	<0.001 (0.742)
Music & Audio	0.236 (0.355)	0.217 (0.344)	0.192 (0.346)
Photo & Video	<0.001 (0.672)	<0.001 (0.672)	<0.001 (0.677)
Business	<0.001 (0.750)	<0.001 (0.740)	<0.001 (0.757)
Maps & Navigation	<0.001 (0.808)	<0.001 (0.806)	<0.001 (0.808)
Sports & Recreation	<0.001 (0.784)	<0.001 (0.785)	<0.001 (0.794)
Finance	<0.001 (0.590)	<0.001 (0.594)	<0.001 (0.604)
IM & Social Networking	<0.001 (0.644)	<0.001 (0.637)	<0.001 (0.647)
News	<0.001 (0.457)	<0.001 (0.470)	<0.001 (0.480)
Weather	<0.001 (0.699)	<0.001 (0.739)	<0.001 (0.774)
Shopping	<0.001 (0.591)	<0.001 (0.611)	<0.001 (0.645)
Category	CBR (worst k) vs.MeanPrice	CBR (mean k) vs MeanPrice	CBR (best k) vs.MeanPrice
Games	<0.001 (0.914)	<0.001 (0.937)	<0.001 (0.946)
Utilities	<0.001 (0.898)	<0.001 (0.906)	<0.001 (0.918)
Entertainment	<0.001 (0.972)	<0.001 (0.981)	<0.001 (0.983)
Travel	<0.001 (0.972)	<0.001 (0.981)	<0.001 (0.983)
Health & Wellness	<0.001 (0.945)	<0.001 (0.950)	<0.001 (0.956)
Education	<0.001 (0.971)	<0.001 (0.971)	<0.001 (0.975)
Productivity	<0.001 (0.855)	<0.001 (0.884)	<0.001 (0.899)
Music & Audio	<0.001 (0.944)	<0.001 (0.947)	<0.001 (0.958)
Photo & Video	<0.001 (0.911)	<0.001 (0.939)	<0.001 (0.947)
Business	<0.001 (0.900)	<0.001 (0.895)	<0.001 (0.896)
Maps & Navigation	<0.001 (0.928)	<0.001 (0.926)	<0.001 (0.923)
Sports & Recreation	<0.001 (0.927)	<0.001 (0.928)	<0.001 (0.938)
Finance	<0.001 (0.895)	<0.001 (0.898)	<0.001 (0.907)
IM & Social Networking	<0.001 (0.844)	<0.001 (0.858)	<0.001 (0.870)
News	<0.001 (0.741)	<0.001 (0.745)	<0.001 (0.783)
Weather	<0.001 (0.740)	<0.001 (0.794)	<0.001 (0.839)
Shopping	<0.001 (0.787)	<0.001 (0.818)	<0.001 (0.850)
Category	CBR (worst k) vs. MedianPrice	CBR (mean k) vs MedianPrice	CBR (best k) vs. MedianPrice
Games	<0.001 (0.827)	<0.001 (0.859)	<0.001 (0.859)
Utilities	<0.001 (0.765)	<0.001 (0.768)	<0.001 (0.779)
Entertainment	<0.001 (0.837)	<0.001 (0.841)	<0.001 (0.843)
Travel	<0.001 (0.847)	<0.001 (0.851)	<0.001 (0.855)
Health & Wellness	<0.001 (0.845)	<0.001 (0.838)	<0.001 (0.856)
Education	<0.001 (0.800)	<0.001 (0.797)	<0.001 (0.803)
Productivity	<0.001 (0.756)	<0.001 (0.778)	<0.001 (0.794)
Music & Audio	0.236 (0.355)	0.217 (0.344)	0.355 (0.352)
Photo & Video	<0.001 (0.717)	<0.001 (0.725)	<0.001 (0.32)
Business	<0.001 (0.755)	<0.001 (0.745)	<0.001 (0.762)
Maps & Navigation	<0.001 (0.893)	<0.001 (0.888)	<0.001 (0.879)
Sports & Recreation	<0.001 (0.792)	<0.001 (0.792)	<0.001 (0.802)
Finance	<0.001 (0.818)	<0.001 (0.815)	<0.001 (0.837)
IM & Social Networking	<0.001 (0.694)	<0.001 (0.720)	<0.001 (0.722)
News	<0.001 (0.613)	<0.001 (0.638)	<0.001 (0.648)
Weather	<0.001 (0.733)	<0.001 (0.796)	<0.001 (0.826)
Shopping	<0.001 (0.758)	<0.001 (0.820)	<0.001 (0.838)

We found that in only 152 out 3570 of cases, a significant difference among the absolute residuals was observed. In the remaining 3418 cases no significant difference was found. To summarise the results of the Wilcoxon comparisons, we use the following win-tie-loss procedure [19]; if the distribution i is statistically significantly better (less) than j according to the Wilcoxon test we updated win_i and $loss_j$, otherwise we incremented tie_i and tie_j. Figure 5 reports the percentage of win-tie-loss values achieved by employing CBR with different numbers of analogies over all categories. This graphically illustrates the difference in relative performance of each of the different choices for k.

Employing five analogies provided us the best balance among win-tie-loss, while the worst is achieved by using 15 analogies. Moreover, we observed that no significant difference was found among the results obtained with the configurations employed on 10 categories (i.e., `Entertainment`, `Travel`, `Education`, `Music & Audio`, `Business`, `Sports & Recreation`, `IM & Social Networking`, `News`, `Shopping`), while very few differences were found on the remaining ones.

TABLE 4
Comparison of the market price strategies (i.e., MinPrice, MeanPrice and MedianPrice) and Random Guessing (RG) using Standardised Accuracy (SA) and Wilcoxon Test (effect size in parentheses).

Category	MinPrice vs. RG		MeanPrice vs. RG		MedianPrice vs. RG	
	SA	Wilcoxon Test	SA	Wilcoxon Test	SA	Wilcoxon Test
Games	11	<0.001 (0.570)	23	<0.001 (0.631)	24	<0.001 (0.758)
Utilities	29	<0.001 (0.769)	26	<0.001 (0.721)	38	<0.001 (0.865)
Entertainment	29	<0.001 (0.790)	25	<0.001 (0.740)	39	<0.001 (0.847)
Travel	24	<0.001 (0.678)	24	<0.001 (0.706)	33	<0.001 (0.840)
Health & Wellness	32	<0.001 (0.783)	16	<0.001 (0.694)	38	<0.001 (0.798)
Education	29	<0.001 (0.780)	24	<0.001 (0.726)	39	<0.001 (0.837)
Productivity	28	<0.001 (0.776)	24	<0.001 (0.745)	37	<0.001 (0.815)
Music & Audio	41	<0.001 (0.808)	15	<0.001 (0.629)	41	<0.001 (0.808)
Photo & Video	22	<0.001 (0.672)	28	<0.001 (0.719)	36	<0.001 (0.920)
Business	35	<0.001 (0.824)	22	<0.001 (0.767)	41	<0.001 (0.856)
Maps & Navigation	22	<0.001 (0.771)	22	<0.001 (0.680)	31	<0.001 (0.792)
Sports & Recreation	25	<0.001 (0.751)	24	<0.001 (0.698)	38	<0.001 (0.839)
Finance	33	<0.001 (0.759)	26	<0.001 (0.722)	35	<0.001 (0.848)
IM & Social Networking	34	<0.001 (0.801)	25	<0.001 (0.758)	39	<0.001 (0.860)
News	29	<0.001 (0.682)	21	<0.001 (0.703)	21	<0.001 (0.744)
Weather	-3	0.642 (0.553)	24	<0.001 (0.707)	15	<0.001 (0.640)
Shopping	23	0.003 (0.617)	24	<0.001 (0.693)	24	<0.001 (0.733)

TABLE 5
Mean Absolute Residuals obtained with CBR employing 1 up to 15 analogies.

	Games	Utilities	Entertainment	Travel	Health & Wellness	Education	Productivity	Music & Audio	Photo & Video	Business	Maps & Navigation	Sports & Recreation	Finance	IM & Social Networking	News	Weather	Shopping
CBR1	0.27	1.07	0.27	0.35	2.00	0.38	2.20	0.29	0.39	5.94	2.42	0.59	1.46	1.40	0.69	2.06	0.73
CBR2	0.22	0.80	0.26	0.25	1.99	0.36	1.90	0.25	0.28	4.94	2.21	0.66	1.47	1.23	0.62	1.99	0.88
CBR3	0.20	0.85	0.28	0.21	2.12	0.37	1.74	0.27	0.22	5.84	2.35	0.55	1.40	1.14	0.66	2.06	0.61
CBR4	0.20	0.85	0.26	0.24	2.18	0.40	1.67	0.26	0.20	6.66	2.40	0.48	1.34	1.11	0.56	2.12	0.57
CBR5	0.19	0.86	0.26	0.22	2.03	0.43	1.67	0.25	0.21	6.23	2.35	0.48	1.33	1.10	0.56	2.30	0.61
CBR6	0.19	0.86	0.25	0.22	2.07	0.42	1.67	0.22	0.19	6.31	2.51	0.48	1.34	1.14	0.53	2.23	0.60
CBR7	0.20	0.89	0.25	0.21	2.29	0.41	1.63	0.21	0.18	6.26	2.49	0.50	1.36	1.12	0.61	2.34	0.75
CBR8	0.19	0.89	0.25	0.21	2.37	0.41	1.63	0.20	0.18	6.21	2.47	0.53	1.38	1.19	0.61	2.39	0.75
CBR9	0.19	0.90	0.24	0.21	2.32	0.41	1.59	0.20	0.17	6.07	2.49	0.54	1.37	1.17	0.65	2.53	0.73
CBR10	0.19	0.91	0.25	0.21	2.27	0.42	1.57	0.19	0.17	6.11	2.48	0.56	1.41	1.14	0.64	2.75	0.79
CBR11	0.19	0.91	0.25	0.22	2.21	0.42	1.59	0.19	0.18	6.16	2.46	0.58	1.43	1.16	0.66	2.70	0.80
CBR12	0.18	0.90	0.26	0.22	2.24	0.43	1.63	0.20	0.19	6.21	2.51	0.59	1.46	1.18	0.72	2.68	0.83
CBR13	0.18	0.90	0.25	0.22	2.27	0.45	1.61	0.19	0.19	6.26	2.59	0.67	1.48	1.20	0.75	2.77	0.88
CBR14	0.18	0.91	0.25	0.22	2.24	0.45	1.61	0.20	0.20	6.31	2.67	0.68	1.49	1.21	0.75	2.91	0.94
CBR15	0.18	0.91	0.26	0.21	2.27	0.47	1.63	0.19	0.20	6.29	2.63	0.70	1.50	1.22	0.76	3.00	0.94
Min	0.18	0.80	0.24	0.21	1.99	0.36	1.57	0.19	0.17	4.94	2.21	0.48	1.33	1.10	0.53	1.99	0.57
Avg	0.20	0.89	0.26	0.23	2.19	0.42	1.69	0.22	0.21	6.12	2.47	0.57	1.41	1.18	0.65	2.45	0.76
Max	0.27	1.07	0.28	0.35	2.37	0.47	2.20	0.29	0.39	6.66	2.67	0.70	1.50	1.40	0.76	3.00	0.94
St.Dev.	0.02	0.06	0.01	0.04	0.12	0.03	0.16	0.03	0.06	0.37	0.11	0.08	0.06	0.07	0.07	0.33	0.12

Thus, in answer to RQ8: a few analogies (five) are sufficient to be effective over all categories.

7 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. In our study, construct validity threats may arise from the assumptions we make about the current state of the art and practice.

Since we are the first to address this problem, there simply is no currently implemented state of the art in terms of predictive price modelling based on app features. We therefore compared our results for technical quality against random guessing; a baseline benchmark to assess the usefulness of a prediction system [9]. We also found comparatively little literature to guide us on what we should consider to be the 'standard practice' adopted by developers to price their apps. Market reports and advice all point towards the Min, Mean and Median market price strategies against which we compared our results.

Concerning conclusion validity, we carefully applied the statistical tests, verifying all the assumptions each inferential test requires concerning the distributions to which it is applied. Our approach to external threats is also 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.

8 Related Work

Recent work on Mining Software Repositories (MSR) has produced scalable techniques for exploring the wealth of source code and associated documentation that can be found in software repositories [20]. This MSR work

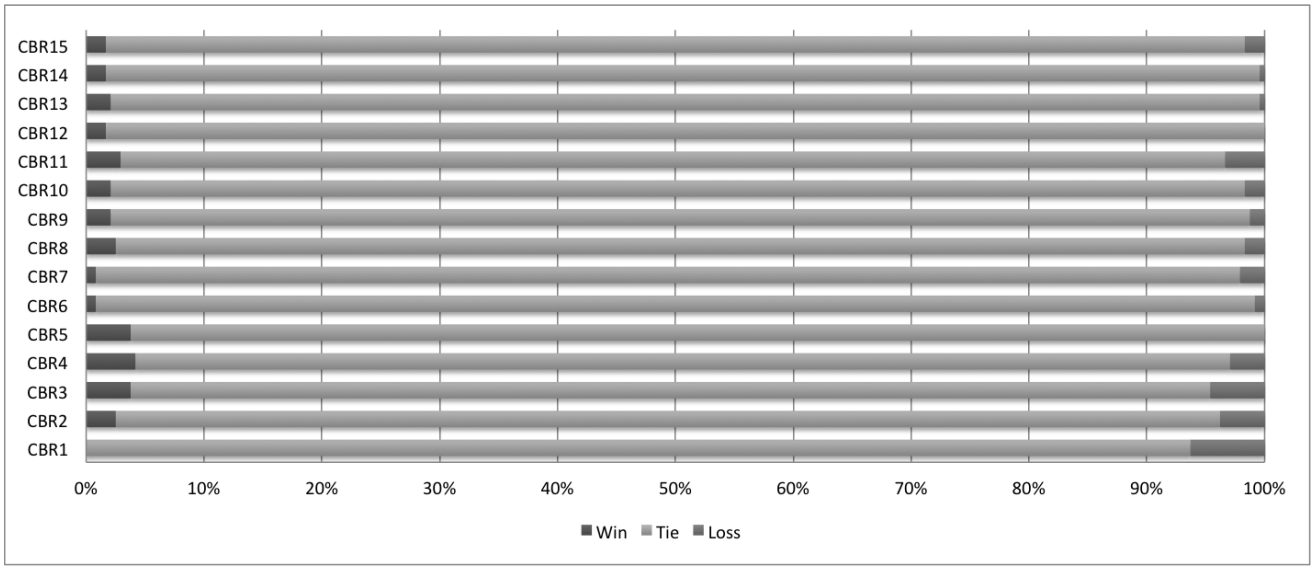

Fig. 5. Percentage of win-tie-loss results from Wilcoxon Test performed on the absolute residuals obtained using CBR with different numbers of analogies on all the considered categories.

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.

In our view an app store is a form of software repository. AppStore mining is therefore a form of Mining Software Repositories (MSR). 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 [21]. Though there has been work on app store analysis [22], we believe that ours is the first paper to data mine and analyse app features and their relationship to non-technical information.

Previous work on MSR has tended to focus on understanding, predicting and, ultimately guiding and controlling the process of software evolution [23], [24]. 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 [25] of software products of (sometimes) very large size [26]. 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.

An app store called Google Play (formerly named as the Android Market) [30] was opened in October 2008. There were 700,000 apps available with over 25 billion downloads as of September 2012 [31]. BlackBerry App World [32] opened in April 2009. There have been more than 99,500 available apps in the store for three years [33]. 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 value of the findings remains considerable.

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 [27]. Since then, there has been a dramatic increase in terms of the number of functionalities and platforms for the smartphones. The largest app store is created by Apple Inc [28] and initially released in July 2008. There were over 775,000 apps in total as of Jan 2013 [29]. Over 40 billion apps have been downloaded in the four years that the Apple App Store has been open.

Until recently, there has been little work [1][22][41][42] on app stores as sources of software engineering data. Yamakami [42] proposes the notion of 'open source based mobile platform software engineering' and discusses a business model between software vendors, carriers and handset vendors. Want [41] highlights how the app store allows developers to integrate, test and distribute the key aspects of pervasive computing research. Minelli and Lanza [43] carry out an investigation on Android apps based on source code, usage of APIs and historical data to examine the differences compared with traditional software systems. Lulu and

Kuflik [44] use machine learning approach to clustering apps based on their functionality extracted from the app description. Menzies [45] also points out that app store data can be used for software engineering data mining. Iacob and Harrison [46] introduce a prototype to extract feature requests from mobile app online reviews to assist app developers. Pandita et al. [47] present a framework WHYPER to examine the permissions needed in an mobile app description by using NLP techniques in order to support the app risk assessment.

None of this previous work (apart from the earlier version of this paper [1]) uses the *features* of the apps residing in the app store studied. We argue that the incorporation of technical information (such as feature information, mined from app descriptions) is important to better understand the relationships between technical, business and social aspects of app store ecosystems.

9 Actionable Findings and Future Work

Our findings from the first three research questions are potentially important for both researchers and for developers of apps. The findings tell developers that the ratings that accrue to apps do, in general, correlate with apps' popularity. This suggests that developers would be wise not to ignore the users' evaluations of their apps. For researchers, this finding also provides a motivation and stimulus for further research on data mining of user evaluations, in order to better understand customer sentiment. There is evidence to suggest that this uptake in mining user evaluations is already taking place [46]. Of course, our findings are based on a single snapshot of the Blackberry App Store. We hope that our findings are also a motivation for future work on other apps stores and for more longitudinal studies of the correlations between aspects such as price, popularity and customer sentiment.

The results of our fourth research question have potential actionable findings for researchers since they provide evidence to support claims that the extracted feature information is meaningful and can be used to understand the relationships between the features of an app and its popularity and ratings.

This could provide a 'missing piece' of the overall jigsaw of information available to researchers studying app stores. That is, we have readily available ratings and pricing information, but we do not always have source code available; the technical information about an app is this critical 'missing piece', without which we cannot connect the technical and business aspects of app stores. Our results from RQ4 indicate that this information can be mined from app descriptions, opening the possibility to consider apps' features as objects of study in their own right.

There are many potential avenues for future work that result from this finding. 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 users to better navigate and select the apps most suited to their needs. Current categories are defined in a rather arbitrary manner and some, such as games, productivity and business app categories are in need of further sub-categorisation. Such a clustering approach, based on features, might provide a more semantic justification for categorisation and also help to identify emergent sub categories.

In the final three research questions (RQs 5, 6 and 7), we explore some of the possibilities open to research in app store analysis, using the extracted feature information. We believe that our results in this part of the paper have merely 'scratched the surface' of what is possible, but we hope that they will prove sufficient to illustrate the potential of this new area for the 'Mining Software Repository' community.

We also believe that these research questions have potentially actionable implications for the app developer community. More specially, our results from RQs 5, 6 and 7 provide evidence that supports the claim that extracted feature information is sufficiently reliable to be the basis of a price prediction system that outperforms current market price strategies. These findings may have actionable implications for app store developers in general and for those working within the Blackberry App World market in particular. Perhaps our finding (RQ7) that three existing market price practices outperform random guessing offers, at least, some consolation and comfort to developers faced with the challenge of finding a suitable price for their products. However, since current advice to app store developers [11] is to select the mean price within a category as a suitable 'guesstimate price', there is clearly room for improvement; this current approach degenerates to random guessing over repeated trials.

This app pricing problem is very unlike, for example, bespoke software application pricing problems. When an organisation prices a bespoke software development task for a single customer there are, of course, natural challenges in determining the requirements and ensuring customer satisfaction. Software engineers developing apps reside at the other end of the software granularity scale: app developers sell enormous numbers of units (at comparatively low prices) compared to bespoke development organisations, which sell only one, or a few, applications (at a comparatively high price).

The typical price charged for an app in some of the most popular app store categories is between one and five dollars, yet such an app may sell hundreds of thousands or even millions of units. A poor choice of

price that is misestimated by as little as $0.50 could result in revenue loss of several millions of dollars. It is therefore crucial that app providers charge the right price for their apps; the survival of the organisation may depend upon it, with microscopic price fluctuations having enormous potential impact on the organisation's bottom line.

One might suppose that the wily app store developer could start off with an initial launch price that is low (in order to attract business), subsequently increasing the price as the app becomes more popular. Unfortunately this apparently cautious approach usually fails; evidence indicates that markets are resistant to such incremental price creep [48]. Alternatively, the developer might start at a very high price, seeking to discount until 'the price is right'. Sadly, once again, the evidence suggests that markets do not respond well to initial overpricing [48].

Fortunately, our other findings (RQ5 and RQ6) provide evidence to suggest that the combination of data mining and machine learning introduced in the paper can significantly outperform current price predicting practices. Also, our price prediction system is based on the simple (and intuitive) approach of case based reasoning. Unlike other more complex machine learning techniques, case based reasoning has relatively few parameters that require tuning.

Furthermore, our finding (RQ8) that five analogies are suitable for almost all of the app store categories is promising; it provides a usable default parameter setting. The implications for practising app developers is that the prediction system will require mercifully little tuning and can be deployed immediately. Ease of deployment is an important property because developers have neither the time nor the inclination for the tedious and error-prone task of parameter tuning.

Of course, we do not propose that app developers would simply use our approach as the only way in which they would go about pricing their apps. Rather, we believe that our approach could provide a useful guide to pricing policies; acting as a sanity check and a guide to the likely market value of a feature. The approach might also help to identify features that attract a higher price so that these could be prioritised in development. In future work we plan to consider the prediction of popularity based on features, which might also be useful in prioritising features for development and enhancements.

The scientific evidence we present in this paper is based upon the Blackberry App World market. Naturally, we cannot (and do not) claim that these findings will necessarily generalise to other markets and platforms. However, our results offer the prospect that it will prove possible to develop suitable app price prediction systems for other app platforms. The investigation of such systems remains an open problem for future work by the research community.

There also remain many other challenging (but exciting) open problems for App Store Mining. Previous work on predictive modelling for software engineering has been largely confined to the two areas of bug prediction [49][50] and effort estimation [51][52].

With this paper we seek to demonstrate that predictive modelling can be applied in the new and rapidly developing world of app stores. In order to facilitate and support the research community in developing this research agenda, we make available all data on the apps and their features and properties reported in this paper. The datasets used in the work reported in this paper can be downloaded from the UCLAppA page:

`www0.cs.ucl.ac.uk/staff/Y.Jia/projects/app_store_mining/`

In future, we 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) [53][54].

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 intend to explore the migration of features between apps and the degree to which our findings also apply to other app stores.

As can be seen from foregoing discussion the rich and varied sources of information available to researchers from app stores and the products they contain open up many exciting avenues for future research; we cannot explore all of these avenues and hope that this paper will stimulate the wider MSR research community to join us in taking up this challenge. We will continue to provide data from our work to support and facilitate research community uptake.

10 Conclusions

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

We have introduced a method to extract, from app store descriptors, usable feature information that captures some of the technical character of the apps in the store. We evaluated our approach on the non-free apps in the Blackberry App Store. Our results indicate that the features we extract using data mining can maintain correlations between app characteristics (ratings and popularity) and that we can construct competitive price estimates using the extracted feature information. We believe that these results demonstrate the value of app store analysis and open up a potentially rich avenue for future Software Repository Mining research.

ACKNOWLEDGMENT

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] M. Harman, Y. Jia, and Y. Zhang, "App Store Mining and Analysis: MSR for App Stores," in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR '12)*. Zurich, Swiss: IEEE, June 2012, pp. 108–111.

[2] E. Loper and S. 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)*. Philadelphia, USA: Association for Computational Linguistics, 7-12 July 2002, pp. 69–72.

[3] M. Shepperd and C. Schofield, "Estimating Software Project Effort using Analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736 –743, Nov 1997.

[4] L. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. Maxwell, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques," in *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. Los Angeles, California, USA: IEEE, 16-22 May 1999, pp. 313 –323.

[5] E. Mendes, "The Use of Bayesian Networks for Web Effort Estimation: Further Investigation," in *Proceedings of the 8th International Conference on Web Engineering (ICWE '08)*. Yorktown Heights, NJ, USA: IEEE, 14-18 July 2008, pp. 203 –216.

[6] L. Briand, T. Langley, and I. Wieczorek, "A replicated Assessment and Comparison of Common Software Cost Modeling Techniques," in *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. Los Angeles, California, USA: IEEE, 16-22 May 1999, pp. 313–322.

[7] F. Ferrucci, E. Mendes, and F. Sarro, "Web Effort Estimation: The Value of Cross-company Data Set Compared to Single-company Data Set," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE '12)*. Lund, Sweden: ACM, 21-22 September 2012, pp. 29–38.

[8] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, October 2009.

[9] M. J. Shepperd and S. G. MacDonell, "Evaluating Prediction Systems in Software Project Estimation," *Information & Software Technology*, vol. 54, no. 8, pp. 820–827, August 2012.

[10] ImpigerMobile, "How to Price my iPhone app?" [Online]. Available: http://www.impigermobile.com/how-to-price-my-iphone-app/

[11] P. Viswanathan, "How to Price Your Mobile Application." [Online]. Available: http://mobiledevices.about.com/od/carrierfaq/ht/How-To-Price-Mobile-Application.htm

[12] Apppli, "The essentials of mobile app marketing, white paper." [Online]. Available: http://www.apppli.com/whitepapers/

[13] Pratt, "Finding the Right Price - Marketing Mobile Software." [Online]. Available: http://www.creativealgorithms.com/blog/?q=content/finding-right-price-marketing-mobile-software-?-part-ii-price

[14] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*, 2nd ed. Routledge Academic, Jan. 1988.

[15] J. P. Royston, "An Extension of Shapiro and Wilk's W Test for Normality to Large Samples," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 2, pp. 115–124, 1982.

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

[17] A. Arcuri and L. Briand, "A Practical Guide for using Statistical Tests to Assess Randomized Algorithms in Software Engineering," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. Hawaii, USA: ACM, 21-28 May 2011, pp. 1–10.

[18] L. D. Mueller and L. Altenberg, "Statistical Inference on Measures of Niche Overlap," *Ecology*, vol. 66, no. 4, pp. 1204–1210, 1985.

[19] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the Value of Ensemble Effort Estimation," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1403–1416, 2012.

[20] A. E. Hassan, "The Road Ahead for Mining Software Repositories," in *Proceedings of the Interlational Conference on Frontiers of Software Maintenance (FoSM '08)*. Beijing, China: IEEE, 28 Sept.-4 Oct. 2008, pp. 48–57.

[21] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli, "On-demand Feature Recommendations Derived from Mining Public Product Descriptions," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. Hawaii, USA: ACM, 21-28 May 2011, pp. 181–190.

[22] B. Eaton, S. Elaluf-Calderwood, C. Sørensen, and Y. Yoo, "Dynamic Structures of Control and Generativity in Digital Ecosystem Service Innovation: The Cases of the Apple and Google Mobile App Stores," The London School of Economics and Political Science, Working Paper Series 183, April 2011.

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

[24] A. Zaidman, B. V. Rompaey, S. Demeyer, and A. 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)*. Lillehammer, Norway: IEEE, April 2008, pp. 220–229.

[25] N. Gruska, A. Wasylkowski, and A. Zeller, "Learning from 6,000 Projects: Lightweight Cross-project Anomaly Detection," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '10)*. Trento, Italy: ACM, 12-16 July 2010, pp. 119–130.

[26] W. Shang, B. Adams, and A. E. Hassan, "Using Pig as a Data Preparation Language for Large-scale Mining Software Repositories Studies: An Experience Report," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2195–2204, October 2012.

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

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

[29] 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."
[30] https://play.google.com/store, "Google Play."
[31] http://officialandroid.blogspot.co.uk/2012/09/google-play-hits-25-billion downloads.html, "Android official Blog."
[32] http://appworld.blackberry.com/webstore/, "BlackBerry App World."
[33] http://www.berryreview.com/2012/05/01/99500-blackberry-apps-now-in-app-world-25-playbook apps/, "Berryreview.com."
[34] A. K. Karlson, S. T. Iqbal, B. Meyers, G. Ramos, K. Lee, and J. C. Tang, "Mobile Taskflow in Context: A Screen Shot Study of Smartphone Usage," in *Proceedings of the 28th ACM Conference on Human Factors in Computing Systems (CHI '10)*. Atlanta, GA, USA: ACM, 10-15 April 2010, pp. 2009–2018.
[35] E. Oliver, "The Challenges in Large-scale Smartphone User Studies," in *Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement (MobiSys '10)*. San Francisco, CA, USA: ACM, 15-18 June 2010.
[36] A. Rahmati and L. Zhong, "A Longitudinal Study of Non-voice Mobile Phone Usage by Teens from an Underserved Urban Community," Rice University, Tech. Rep. 0515-09, May 2009.
[37] T. Armstrong, O. Trescases, C. Amza, and E. de Lara, "Efficient and Transparent Dynamic Content Updates for Mobile Clients," in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys '06)*. Uppsala, Sweden: ACM, 19-22 June 2006, pp. 56–68.
[38] A. Shye, B. Scholbrock, and G. Memik, "Into the Wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '09)*. New York, USA: ACM, 12-16 December 2009, pp. 168–178.
[39] J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode, "Rootkits on Smart Phones: Attacks, Implications and Opportunities," in *Proceedings of the 11th Workshop on Mobile Computing Systems & Applications (HotMobile '10)*. Annapolis, Maryland, USA: ACM, 22-23 February 2010, pp. 49–54.
[40] J. Cheng, S. H. Wong, H. Yang, and S. Lu, "SmartSiren: Virus Detection and Alert for Smartphones," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys '07)*. San Juan, Puerto Rico: ACM, 11-14 June 2007, pp. 258–271.
[41] R. Want, "iPhone: Smarter Than the Average Phone," *Pervasive Computing*, vol. 9, no. 3, pp. 6–9, July-September 2010.
[42] T. 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)*. Seoul, Korea: ACM, 24-26 November 2009, pp. 206–211.
[43] R. Minelli and M. 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: IEEE, 5-8 March 2013.
[44] D. L. Ben Lulu and T. 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)*. Santa Monica, CA, USA: ACM, 19-22 March 2013, pp. 297–306.
[45] T. 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.
[46] C. Iacob and R. 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.
[47] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards Automating Risk Assessment of Mobile Applications," in *Proceedings of the 22nd USENIX Security Symposium*, Washington DC, USA, 14-16 August 2013.
[48] T. van Agten, "The impact of price changes." [Online]. Available: http://www.distimo.com
[49] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, Nov.-Dec. 2012.
[50] C. Catal and B. Diri, "A Systematic Review of Software Fault Prediction Studies," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7346–7354, 2009.
[51] M. Jorgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies," *IEEE Transactions on Software Engineering,*, vol. 33, no. 1, pp. 33–53, January 2007.
[52] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic Literature Review of Machine Learning based Software Development Effort Estimation Models," *Information and Software Technology*, vol. 54, no. 1, pp. 41–59, January 2012.
[53] M. 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)*. Timisoara, Romania: ACM, 12-13 September 2010, pp. 1–13.
[54] M. Harman and B. F. Jones, "Search Based Software Engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, December 2001.