Winning the App Production Rally

Ehsan Noei
Queen's University
Kingston, Ontario, Canada
e.noei@queensu.ca

Daniel Alencar Da Costa
Queen's University
Kingston, Ontario, Canada
daniel.alencar@queensu.ca

Ying Zou
Queen's University
Kingston, Ontario, Canada
ying.zou@queensu.ca

ABSTRACT

When a user looks for an Android app in Google Play Store, a number of apps appear in a specific rank. Mobile apps with higher ranks are more likely to be noticed and downloaded by users. The goal of this work is to understand the evolution of ranks and identify the variables that share a strong relationship with ranks. We explore 900 apps with a total of 4,878,011 user-reviews in 30 app development areas. We discover 13 clusters of rank trends. We observe that the majority of the subject apps (i.e., 61%) dropped in the rankings over the two years of our study. By applying a regression model, we find the variables that statistically significantly explain the rank trends, such as the number of releases. Moreover, we build a mixed effects model to study the changes in ranks across apps and various versions of each app. We find that not all the variables that common-wisdom would deem important have a significant relationship with ranks. Furthermore, app developers should not be afraid of a late entry into the market as new apps can achieve higher ranks than existing apps. Finally, we present the findings to 51 developers. According to the feedback, the findings can help app developers to achieve better ranks in Google Play Store.

CCS CONCEPTS

• **Applied computing**; • **Software and its engineering**; • **Theory of computation** → *Theory and algorithms for application domains*;

KEYWORDS

Mobile application, Rank, Application market, Empirical study

ACM Reference Format:
Ehsan Noei, Daniel Alencar Da Costa, and Ying Zou. 2018. Winning the App Production Rally. In *Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '18), November 4–9, 2018, Lake Buena Vista, FL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3236024.3236044

1 INTRODUCTION

App markets, such as Google Play Store [33], are becoming more competitive year by year [58]. When it comes to users' choice when downloading an app, the app ranks play a major role [61]. When a user looks for a particular sort of app (i.e., area), Google Play Store lists the most related apps with respect to the app ranks [61]. Improving the rank of an app increases the chance of being noticed and downloaded by users. Having a higher number of users is desired as it can increase the revenue of app development companies [10, 46].

Prior work estimates app success by studying different factors, such as star-ratings and the number of downloads [58]. The relationship between the star-ratings and various variables, such as the number of functions, has been thoroughly explored by prior research [8, 10, 20, 29, 37, 46, 52, 68, 95]. For example, Bavota *et al.* [10] show the correlation between the star-ratings of an app and the fault- and change-proneness of the APIs that are used by the app. Although star-ratings and the number of downloads may express the success of an app, they might be misleading in some cases. For example, a 5-star app could be downloaded only a couple of times, as opposed to 3-star app with thousands of downloads. Conversely, app ranks clearly express the likelihood that an app will attract more users. Moreover, it is known that Google considers star-ratings and the number of downloads when calculating the ranks [31, 32].

We define *rank trend* as an evolution of ranks (i.e., declines and inclines over time) that happens similarly among a set of apps. We study the rank trends and highlight the variables that are statistically significantly related to the rank trends. In addition, we conduct fine-grained analyses per app and per version of each app over time. Our goal is to find the most important variables that share a significant relationship with the changes in the ranks beside star-ratings and the number of downloads [31, 32]. We investigate 900 apps in 30 different areas that are associated with 4,878,011 user-reviews in two years.

Our findings reveal various practical variables, such as the appearance of new topics, which share a significant relationship with the ranks. Such findings can help both developers of currently published apps and developers of start-up apps by providing actionable guidelines to achieve higher ranks. We address the following research questions:

RQ1) What are the rank trends of mobile apps? We identify rank trends (i.e., evolutions of ranks) by applying a fuzzy clustering [13, 99] on the ranks. We observe that, in some areas, falling in the ranks is more likely to happen than rising in the ranks. Moreover, competing in the areas with dominant high ranked apps (i.e., the apps that maintain their ranks constantly on top), such as *budget*, might be harder than competing in areas with fewer dominant apps.

RQ2) Which variables can improve the rank trends? We apply a regression model to explain the identified rank trends [18, 97]. We identify actionable ways, such as introducing new features, to achieve better ranks over time.

ESEC/FSE '18, November 4–9, 2018, Lake Buena Vista, FL, USA

ACM ISBN 978-1-4503-5573-5/18/11…$15.00
https://doi.org/10.1145/3236024.3236044

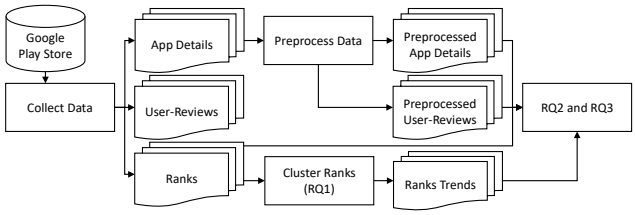

Figure 1: Overview of the study design process.

RQ3) What variables have a significant relationship with the ranks over time? We build a mixed effects model [101] to quantify the ranks across apps and different versions of each app. A mixed effects model allows us to explain the ranks across various trends, areas, and app versions. We observe that developer-related activities, such as release latency, play an important role. We also find that not all the variables that common-wisdom would deem important, such as the number of pictures [90], share a significant relationship with the ranks.

RQ4) How do app developers evaluate the findings? We asked 51 app developers to evaluate our findings. 60.8%, 84.3%, and 80.4% of the developers strongly agree that the findings of each research question can be useful and practical for the industry, respectively. The feedback indicates that higher ranks can be achieved by following our guidelines when developing an app.

The contributions of this paper include:

- We identify the evolution of ranks over time (i.e., rank trends). The majority of the apps that are subject of our study tend to fall in the ranks over time.
- We determine the variables that share a significant relationship with the changes in the ranks. Developers can improve the ranks with respect to our findings.
- We find that not all the variables that may be considered as important variables by developers can statistically significantly explain the changes in the ranks. Therefore, by focusing on our findings, developers can have the rank of an app improved more efficiently.

2 DATA PREPARATION

Figure 1 shows the major steps of the study design process. As shown in Figure 1, we collect the required data, such as release notes and user-reviews, from Google Play Store [33]. Then, we preprocess the collected data.

2.1 Data Collection

Google Play Store has the largest number of apps and users [68]. When it comes to success in the app markets, the reachability of apps speaks first. Therefore, the most effective data collection scenario comes from an ordinary user side. We impersonate an ordinary user by using Google Play Store search engine to find apps, although every user is subject to different attitudes [11, 22].

An area, such as *calculator*, is a subset of apps that share similar purposes and functionalities. We collect the most searched areas using the *Semrush* service [86] which tracks the popular search engines, such as the *Google* search engine. We look for the top keywords that are mostly searched via the mobile devices for finding apps. Table 1 shows the list of 30 top searched keywords (i.e., top wanted areas). The list is determined by manually removing brand names, such as *"Amazon"* and porn-related keywords. By removing brand names, (i) we can focus on the areas that share similar applications, purposes, and functionalities and (ii) we mitigate the skewness of the collected apps towards certain apps.

Table 1: List of the areas with the number of versions and user-reviews. There are 30 apps in each area.

Area	#Versions	#User-Reviews	Area	#Versions	#User-Reviews
Airline	422	51,408	**Health**	263	88,306
Bible	339	92,075	**Mailbox**	339	56,391
Budget	206	18,356	**Messaging**	525	230,248
Calculator	294	7,316	**Movie**	279	164,195
Calling	658	210,483	**News**	690	186,827
Camera	466	55,383	**Paint**	122	6,674
Chess	186	22,811	**Piano**	202	27,909
Cloud	546	162,591	**Radio**	504	238,336
Coupon	348	61,160	**Reminder**	292	108,461
Dating	916	165,869	**Sleep**	159	42,713
Dictionary	292	74,073	**Spy Phone**	162	27,328
Emoji	101	4,105	**Talking Pet**	235	31,662
Fitness	272	84,436	**Translator**	206	36,375
GPS	630	151,844	**Weather**	845	178,968
Grocery List	264	50,944	**Weight Loss**	278	58,590

Like an ordinary user, who uses the search bar to find the desired app, we search each area in Google Play Store. Since the majority of queries are generated by users, we searched the exact identified area names in Google Play Store. We scroll down each search result to the end. For each area, we find a median of 250 app. We randomly select 30 apps from each area to perform our analysis. Considering a higher number of apps (i.e., more than 30 for each area) was not feasible in this study because (i) not all the areas have over 30 different apps, and (ii) not all the apps can be studied for two years since some apps are periodically cleaned up from the store [84]. Having 30 apps for each of our 30 subject areas allows us to investigate 900 apps. Moreover, our random selection provides a better combination of apps than picking only the top apps.

We retrieve the app details (e.g., releases notes and descriptions), the user-reviews, and the ranks of the selected apps from Google Play Store. We implemented a crawler to automatically fetch the information of our subject apps on a daily basis for two years (since *March* 1, 2015 to *March* 1 2017). We retrieved all the details of our subject apps and the associated user-reviews. We collect 4,878,011 user-reviews for 10,508 distinct versions of our 900 apps. The number of app versions and informative user-reviews (see Section 2.2) are listed in Table 1 for each area.

2.2 Preprocessing Data

In this subsection, we explain all the steps that are taken to preprocess the app details and the user-reviews.

Removing Non-English User-reviews. We remove 232,286 (i.e., 5%) user-reviews that were written in a non-English language using Language Detector [70].

Filtering Out Uninformative User-reviews. Users do not always leave an *informative* review. For instance, *"OK app"* provides minor information for app developers [17, 96]. We rely on the AR-Miner [17] to filter out uninformative user-reviews. In total,

we identified 2,508,691 informative user-reviews (i.e., 54% of the English user-reviews). Table 1 shows the number of informative user-reviews in each area.

Correcting Typos and Informal Vocabularies. User-reviews potentially suffer from typos that can taint the results of text analysis techniques [69]. We fix the typos in user-reviews, app descriptions, and release notes using Jazzy Spell Checker [44] with a dictionary of 645,289 English words. Based on manually investigating 384 user-reviews with a confidence level of 95% and a confidence interval of 5, Jazzy corrects 62% of incorrect words. In addition, we replace the abbreviations and informal vocabularies that are commonly used by users [4, 65] with the right words. We retrieve the abbreviations and informal vocabularies from the available online sources [4, 65].

Breaking Down App Descriptions and Release Notes. Unlike user-reviews, app descriptions and release notes benefit from a structured form of writing. Google Play Store demonstrates app descriptions and release notes using the standard HTML format [33, 82]. We extract each item from each release note. We also break app descriptions into separate paragraphs and items of the existing lists in the descriptions. Organizing app descriptions and release notes allows us to measure the variables more accurately (see Table 2).

Resolving Synonyms. General-purpose thesaurus, such as WordNet [62], is not adequate to resolve the synonyms of informal texts, such as user-reviews [96]. We build a dictionary of words by manually studying 5,000 randomly selected sample user-reviews, app descriptions, and release notes [96]. From each set of synonyms, we pick one as the representative word. For example, *great*, *awesome*, and *dope*, belong to the same group of words.

Resolving Negations. Negations in user-reviews, app descriptions, and release notes can disturb the text processing techniques. For example *"not good"* should be replaced by *"bad"*. To avoid such an issue, we use the Stanford natural language processing toolkit [56] to resolve the negated terms [96].

Topic Modeling. We apply topic modeling [2] on app description, release notes, and user-reviews for two reasons. First, we would be able to unify all the user-reviews, app descriptions, and release notes within a certain number of topics and assign mathematically comparable vectors to each user-review, app description, and releases note. Therefore, we could measure the introduction of new topics and the similarity of release notes and app descriptions with user-reviews (see Table 2). Second, using topic modeling helps us to mitigate the lack of up-to-date dictionary of words for the terms that are used by users [68]. Similarly, Noei and Heydarnoori [68] proposed a tool, called EXAF, that helps developers find sample applications that implement a desired framework-provided concept. EXAF employs topic modeling for finding the software engineering terms that represent the same concepts.

We build a corpus using user-reviews, app description, and release notes. Before building the corpus, we remove the stopwords [81] and stem the words [54]. For example, *"installed"* and *"installing"* have the same word stem which is *"install"*.

We apply the Latent Dirichlet Allocation (LDA) technique [14, 67] on the user-reviews, app descriptions, the release notes. Each document in the corpus is considered as a combination of a number of topics [14]. Prior to the topic modeling, LDA parameters need to be set up. α is the parameter on the per-document topic distributions, and β is the parameter on the per-topic word distribution [14]. Furthermore, we employ three latest approaches (i.e., Griffiths *et al.* [35], Deveaud *et al.* [24], and Cao *et al.* [15]) to estimate the optimum number of topics. The summary of the outputs of the three approaches (i.e., Griffiths, Deveaud, and Cao) suggests that the optimum number of topics is between 25 and 165. To avoid losing any potential topic, we pick 165 (i.e., maximum) as the optimum number of topics. We run the LDA with 2,000 Gibbs sampling iterations [34, 80]. With 2,000 iterations, we can achieve a high accuracy of LDA [35, 80].

2.3 Measuring Variables

We use the *Goal / Question / Metric* (GQM) paradigm [9, 92] to capture the required variables. Our goal is to understand ranks and rank trends using the available data. It is known that Google considers star-ratings and the number of downloads when calculating the ranks [31, 32]. Therefore, we consider star-ratings and the number of downloads as *control variables* in our study. Table 2 shows the list of our 45 variables, followed by a motivation, a brief description, and the number of sub-variables for each variable.

3 APPROACH AND RESULTS

In this section, we present the approach and the findings of the research questions.

RQ1) What are the rank trends of mobile apps?

Motivation. When searching for new apps, Google Play Store shows a ranked list of apps. Apps with higher ranks are more likely to be downloaded and installed [21, 51, 100]. Observing the evolution of ranks helps developers to see the odds of rising and falling in ranks over time. Moreover, we can discover the areas where apps tend to change or tend to maintain their ranks. Thus, developers can take a wiser decision when implementing a new app or trying to improve the ranks of the currently published apps.
Approach. We apply time series clustering [49] to identify the rank trends. Time series clustering requires (i) a clustering algorithm, (ii) a distance measurement method, and (iii) the number of clusters. We apply a fuzzy clustering algorithm [99] with dynamic time warping (DTW) [12] as the method of measuring the distance between the time series of ranks.

Fuzzy Clustering. Partition clusterings, such as k-means [98], build n partitions where each cluster contains at least one object and each object belongs to one cluster. Fuzzy clustering inherits from the fuzzy logic and fuzzy sets where the truth values of variables can be real numbers between 0 and 1 instead of being exactly 0 or 1 [99]. Therefore, with fuzzy clustering, a rank trend can belong to more than one cluster [13]. For example, if a rank is fluctuating while declining over time, the fuzzy algorithm can put such a rank into two clusters. Therefore, the variables associated with such a trend can contribute to both clusters of ranks. The fuzzy algorithms can be set with a level of fuzziness m where $m \geq 1$. A larger m makes the clustering fuzzier, while when $m = 1$, the fuzzy algorithm works like a partition algorithm [49].

Dynamic Time Warping (DTW). DTW aligns two time series (rank trends) in a way that the differences between the two trends are minimized [12]. The Euclidean distance that is commonly used

Table 2: Details of the independent variables, including motivations, descriptions, and the numbers of variables.

Variable(s)	Motivation (m) | Description (d)	Count
Star-Ratings	**m.** Google Play Store provides a star-rating mechanism with which users can rate each app from 1 star (worst) to 5 stars (best) [33]. We consider star-rating in our study as a control variable because Google uses star-ratings as one of the factors to calculate the ranks [31, 32]. **d.** We collect the star-ratings that are associated with each version of the subject apps.	1
#Downloads	**m.** We measure the number of downloads as a control variable as Google uses the number of downloads to calculate the ranks [31, 32]. **d.** For each version of an app, we retrieve the latest number of downloads from Google Play Store.	1
#User-Reviews	**m.** The number of user-reviews can represent the interest of users in an app [68]. **d.** For each version of an app, we count the number of user-reviews since the last version to the current version.	1
Sentiment Scores	**m.** As the interpretation of different users is not the same regarding the number of stars [68], measuring the sentiment scores of the user-reviews is an alternative to measure the user-perceived quality. **d.** We measure the average of sentiment scores of the user-reviews for each version using the SentiStrength-SE tool [43]. Unlike star-ratings that are explicitly given by end users, sentiment scores cannot be measured directly from Google Play Store. We manually analyzed the output of the SentiStrength-SE on a sample of 384 user-reviews with the confidence level of 95% and the confidence interval of 5. SentiStrength-SE gives 71% correct sentiment scores.	1
Proportion of User-Reviews	**m.** To capture the diversity of user-reviews, we measure the proportion of negative, positive, and neutral user-reviews from two perspectives: (i) star-ratings, and (ii) sentiment scores. **d.** Users usually do not install the apps with a star-rating of less than 3 [68]. We consider user-reviews with star-ratings equal to 3, greater than 3, and less than 3 as neutral, positive, and negative user-reviews, respectively [68].	6
App Price	**m.** The price of an app can encourage or dissuade a user to download an app [30]. **d.** We catch the price of each app from Google Play Store.	1
#Releases	**m.** The process of changing the code, integrating with the older versions, and releasing as a new version is called release engineering [1]. The higher number of releases can indicate an active release cycle. **d.** We collect the number of releases for each app.	1
Release Latency	**m.** Different apps might have differing release cycles [1] that may affect the ranks [40]. **d.** We measure the time between two releases as an indicator of the release cycle latency for each version. We also measure the average of latencies for each app.	2
Text Size	**m.** Larger app descriptions can give users more information about an app, while users may be interested in brief descriptions and release notes. **d.** We measure the following variables to capture the size of app details: (i) the number of words of app names, (ii) the number of words of app descriptions, (iii) the number of sentences of app descriptions, (iv) the number of words of release notes, and (v) the number of sentences of release notes. Moreover, the size of a user-review can indicate its helpfulness [47]. For each version of an app, we calculate the average of (vi) the number of words and (vii) the number sentences of the user-reviews that are posted for the corresponding version. We use the Stanford Parser and Tokenizer [23, 56] to count the number of words and sentences.	7
App Details Similarity	**m.** The first impression of an app might lie upon its name and description. A relevant name and description can motivate more users to download an app. **d.** We measure the similarity of app names and app descriptions with our 30 subject areas using the WordNet similarity package [76].	2
#Pictures	**m.** Every app on Google Play Store can be associated with some pictures [33]. The pictures may influence the users' decision. **d.** We count the number of pictures on the page of each app.	1
Installation File Size	**m.** Users may refrain to download a big application due to network and storage limitations [68]. **d.** For each version of an app, we collect the size of the installation file.	1
Launch Date	**m.** An app that has been released prior to other competitors may have a better chance to get the attention of users. **d.** We measure the first release date of each app.	1
Addressing User-Reviews	**m.** Recent studies investigate the importance of user-reviews in the deployment of mobile apps [28, 29, 42, 71, 72]. Some researchers focus on the user-reviews [17, 74, 96] to help developers in the process of app deployment. Therefore, we quantify the degrees to which app developers address the user-reviews. **d.** For each version of apps, we measure the cosine similarity of the app description with the user-reviews that are posted from the last version to the current version. We also measure the similarity of release notes with the corresponding user-reviews [45].	2
Introduction of New Topics	**m.** Different apps may introduce new features and functionalities in their new versions. We estimate the addition of new functionalities by counting the number of topics that are added to a newer version of an app in comparison with the preceding versions. **d.** For each version, we count (i) the number of added topics to the description [45], (ii) the number of removed topics from the description, (iii) the number of added topics to the release note, and (iv) the number of removed topics from the release note.	4
Company	**m.** A reputation of a company may impact all the apps that belong to the same company. For example, having a successful app can motivate users to try other products. **d.** For each company, we measure the number of published apps on Google Play Store, star-ratings of the apps, the number of downloads of the apps, the number of paid apps, the average price of the paid apps, and the ratio of paid app per total apps.	6
Area	**m.** An area with plenty of dominant apps might be hard to compete in. This can influence other apps within the same area. **d.** For each app, we consider its area of development (see Table 1).	1
Category	**m.** A category in which an app is released might affect its rank. Some categories may be very competitive with a large number of popular apps. **d.** For each app, we retrieve its category from Google Play Store. In addition, for each category, we measure the number of published apps on Google Play Store, star-ratings of the apps, the number of paid apps, the average price of the paid apps, and the ratio of paid apps per total apps.	6
		Total: 45

in clustering algorithms, such as k-means [55, 98], assumes that the i^{th} point in one trend should be aligned with the i^{th} points of other trends. DTW alignment allows the offsets in the rank trends to be varied [85]. For example, suppose two rank trends $R_1 = \{r_{11}, r_{12}, ..., r_{1n}\}$ and $R_2 = \{r_{21}, r_{22}, ..., r_{2m}\}$, where n is the number of time points in R_1 and m is the number of time points in R_2. If the trends are very similar but with an offset of d days, Euclidean distance fails to converge. However, DTW can fit such trends in one cluster. Equation (1) shows the distance of warping path between two trends.

$$dist_{DTW} = \frac{\sum_{i=1}^{\kappa} \omega_i}{\kappa} \quad (1)$$

DTW build a warping path $W = \{\omega_1, \omega_2, ..., \omega_\kappa\}$ where κ denotes the number of points in W and $max(m, n) \leq \kappa \leq m+n-1$ [49].

Number of Clusters. Prior to applying a clustering algorithm, it is required to determine the number of clusters to obtain the highest clustering quality [63]. To calculate the number of clusters, we follow two approaches. First, we run the clustering algorithm with different numbers of clusters from 2 to 50 and visually inspect the results until we achieve crisp distinguishable clusters [94]. We find 13 as the best number of clusters. Second, we employ the gap statistic approach [91] to estimate the optimum number of clusters. The gap statistic uses the output of the clustering algorithm and compares it with the change in a within-cluster dispersion. The procedure tries different numbers of clusters to maximize the gap statistic value. We apply the gap statistics algorithm with the number of clusters ranging from 2 to 50. The best result achieved with 13 clusters. Both approaches (i.e., visual inspection and statistical analysis) suggest 13 as the optimum number of clusters.

Findings. In this section, we explain our findings and observations regarding clustering the rank trends.

Observation 1_a***: There are 13 rank trends and three general trends.*** Figure 2 shows the centroids of the 13 identified rank trends. As it is shown in Figure 2, three general trends among the rank trends can be observed: (i) falling, (ii) rising, and (iii) maintaining. In

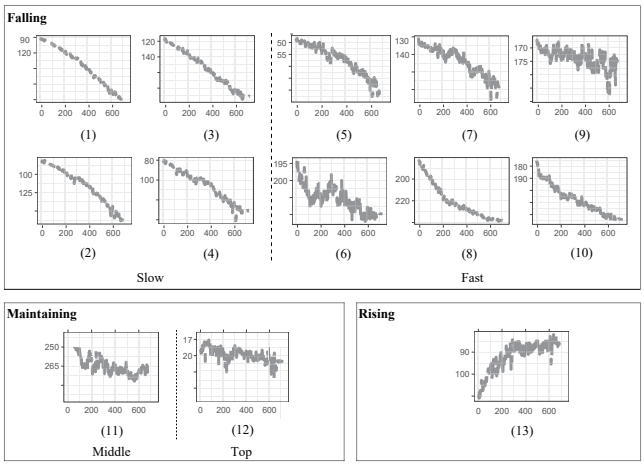

Figure 2: Centroids of each cluster of rank trends. X-axis denotes the time (day) and Y-axis shows the ranks.

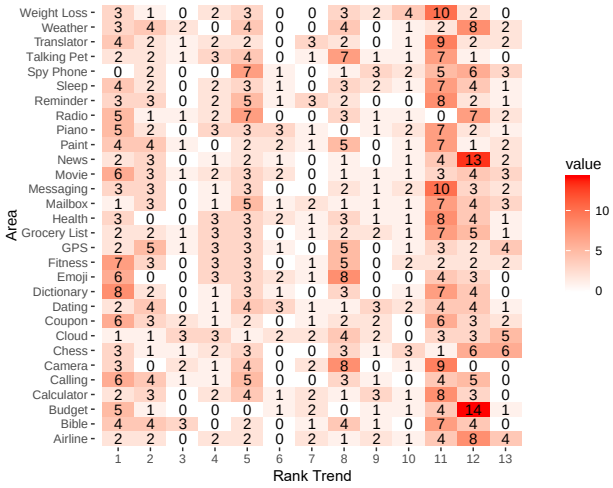

Figure 3: Distribution of apps in each cluster of rank trends (X-axis) versus areas (Y-Axis). A darker shaded red indicates a higher number of a cluster in a given area.

our experiments, we tried increasing the fuzziness of the clustering. However, rank trends tend to stay categorized in only one cluster. This can confirm that the number of clusters is set optimally.

Observation 1_b**: 61% of the studied apps fell in the rankings over time.** By comparing the number of apps that lay on each of the three general trends, we note that the majority of the apps (i.e., 61%) had a falling trend, which may implicitly denote the competitive nature of app markets. Only 6% of the subject apps could have improved the ranks during the studied period. Finally, 33% of our subject apps have maintained their ranks with very slight fluctuations.

Observation 1_c**: Some areas, such as *news* and *budget*, are harder to compete than other areas.** Figure 3 shows the distribution of the rank trends for each area. *News* and *budget* are two areas with the highest number of apps that could have maintained their ranks among the top apps (i.e., cluster 12), while only 3 apps improved their ranks in the *news* and *budget* areas. This observation can suggest that it is harder for newcomers to compete in such areas since there are dominant apps that could have maintained the ranks during the studied period.

Cluster 11 contains the most number of apps (i.e., 19%) in comparison with other clusters. The apps that are in cluster 11 maintained the ranks in the middle. The *radio* apps failed more in maintaining the ranks in comparison with other areas. On the other hand, the *chess* and *cloud* areas have the most number of apps with a rising trend. Such a result suggests that entering into *chess* and *cloud* areas might be a wiser choice to succeed in the competitive market of mobile apps.

> *Most of the subject apps tend to fall in the ranks within the duration of our study. In some areas, such as chess, it is easier to compete and improve the ranks. However, having a large number of dominant apps in some areas, such as news and budget, makes it harder for developers of new apps to compete with the existing apps.*

Table 3: List of correlated variables.

Selected Variable	Correlated Variables
Description Similarity	#Words in Description
#User-Reviews	#Downloads, Average #Downloads (Company), #Negative, Positive, and Neutral User-Reviews, #User-Reviews with Negative, Positive, and Neutral Sentiment Scores
Average Rating (Company)	#Apps (Company)
Address User-Reviews (Release Note)	Address User-Reviews (Description)
Average Price (Company)	#Paid Apps (Company), Price
#Releases	Average Time Between Releases
#Apps (Category)	#Paid Apps (Category)

RQ2) Which variables can improve the rank trends?

Motivation. In **RQ1**, we identified the rank trends. Although one could speculate that the area is an important variable to explain the rank trends (see Figure 3), it is unclear what variables share a strong relationship with the rank trends. Discovering such variables is important to help developers and app development companies to improve the ranks of their apps proactively.

Approach. We treat variables as (i) nominal variables that include two or more categories, such as the category in which an app is published, (ii) ordinal variables that are nominal variables for which the order of values matters, and (iii) numeric variables, such as the number of releases. We also normalize the independent variables to bring the values into the same scale. Before building a model, we identify the correlated variables. Having correlated variables negatively affects the results of our model [39]. We apply variable clustering analysis [38] to build a hierarchical overview of the correlation between the independent variables [68]. We choose the variables that are more intuitive for inclusion in our model from each sub-hierarchy of variables with Spearman's $|\rho| > 0.7$ [66]. The correlated variables are listed in Table 3.

Studying rank trends to get practical results requires ordering them from the worst to the best. However, there is no standard

definition of the goodness of a rank trend to treat the trends as an ordinal variable. Therefore, we mitigate the bias of judging the superiority of each rank trend over another by breaking the trends into two groups of optimistic rank trends and pessimistic rank trends. We define optimistic rank trends as (i) the ranks that are maintained among top apps during our study (i.e., trend 12), and (ii) the ranks with a rising trend (i.e., trend 13). Otherwise, a rank trend is considered as a pessimistic trend.

Having two groups of trends allows us to build an ordered regression model to determine the odds ratios of the variables that play a significant role in explaining the two groups of trends (i.e., optimistic and pessimistic). *Logistic* regression model and *probit* regression model are both appropriate approaches to explain an ordered dichotomous dependent variable (i.e., optimistic vs. pessimistic) [18, 41, 97]. The main differences between *logit* and *probit* regression models are the link function and the assumption of each model about the distribution of the errors [3]. As we are interested to observe the odds ratios, we report the output of the ordered *logistic* regression model in this paper. Nonetheless, both approaches (i.e., *logit* and *probit*) produce the same results with our data. We use variables that are introduced in Table 2 as the independent variables. Our *logistic* regression model obtains a great fit with the McFadden's $\rho^2 = 0.24$ [60]. The McFadden's ρ^2 compares the log-likelihood of the model with the *null* model to measure the level of improvement of the model [26]. Moreover, the number of Events Per Variables (EPV) of our model is 11.5, which indicates that our model has a low risk of over-fitting [89].

We determine the significant variables using the ANOVA χ^2 test [78]. The significant variables have $Pr(>\chi^2)$ less than 0.05. $Pr(>\chi^2)$ is the $p-value$ that is associated with the $\chi-statistic$ [59].

Findings. Table 4 shows the results of our ordered *logistic* regression model. The Likelihood-ratio χ^2 [93], $p-value$, and effect of each independent variables are shown in Table 4. The significant variables are marked with asterisks.

Observation 2_a***:*** **The *launch date* shares a significant relationship with the rank trends.** Interestingly, we observe that the apps that are launched later tend to be more successful in the market. This is good news for newcomers and start-up companies not to be hopeless when launching an app for the first time. Although the initial idea and innovation are essential to succeed [83], competitors might overcome the seminal apps over time if the necessary effort is invested. This observation led us to check how the launch date works when comparing top apps and rising apps only. We observe that the launch date no longer appears as a significant variable when rising apps and top apps are compared.

Observation 2_b***:*** **Developers of published apps should constantly work on improving their apps to strive.** The appearance of new topics (see Section 2.2) in the release notes and the descriptions are two significant variables of the model. A release note usually contains a report of the fixed issues and the addition of new features [33]. A description usually describes the functionality and the purpose of an app [45]. The addition of new topics has a positive relationship with the success of an app.

Observation 2_c***:*** **More releases are encouraged.** Some users may feel frustrated when apps get frequently updated [79]. When it comes to statistical observations, we note that the number of releases has a positive relationship with the success of apps. Therefore, developers should not be afraid of keeping their apps up-to-date. The average of days between releases for the top and rising apps is 14 days. However, developers should be cautious about the quality of the app version they are about to release as Ruiz *et al.* [84] find that releasing a low-quality app endangers the survival of the app.

Observation 2_d***:*** **Developers, especially from start-up companies, should carefully consider the area and the category on which they intend to work.** Both the *area* and the *category* of apps have a significant relationship with the success of an app. A *chess*, *cloud*, *radio*, *spy phone*, *weather*, *news*, or *budget* app has a higher chance to lie upon optimistic trends (by a positive effect). In addition, when we compare only the rising and top maintaining apps, we note that among the aforementioned areas, the ranks of *chess*, *cloud*, *radio*, and *spy phone* apps are more likely to rise.

Table 4: Logistic regression model of rank trends, sorted by p – value. An upward arrow indicates that when the value of the associated variable increases, the rank trend is more likely to fall into optimistic trends.

Variable	LR χ^2	Pr(> χ^2)		Effect
Launch Date	11.94	5.49E-04	***	↗
Appearance of New Topics (Release Note)	10.63	1.12E-03	**	↗
#Releases	7.07	7.85E-02	**	↗
Category	47.69	2.13E-02	*	-
Star-Rating	4.71	3.00E-02	*	↗
Area	44.66	3.18E-02	*	-
Appearance of New Topics (Description)	4.39	3.61E-02	*	↗
Average Rating (Company)	4.32	3.76E-02	*	↗
#User-Reviews	4.04	4.44E-02	*	↗
Description Similarity	3.66	5.57E-02	.	↗
Name Size	3.08	7.92E-02	.	↘
#Pictures	2.91	8.80E-02	.	↗
Address User-Reviews (Release Note)	2.40	1.22E-01		↗
Sentiment Score	2.28	1.31E-01		↗
#Sentences (Description)	1.87	1.71E-01		↘
Average Price (Company)	1.75	1.86E-01		↘
Installation File Size	1.63	2.81E-01		↘
Name Similarity	0.05	8.17E-01		↗
Average Star-Rating (Category)	1.00	9.95E-01		↗
Average Price (Category)	0.00	1.00E+00		↘
#Apps (Category)	0.00	1.00E+00		↘

$p-value$ codes: '***'< 0, '**'< 0.001, '*'< 0.01, '.'< 0.05

App developers should not be afraid of a late entry into the market as newer apps achieved higher ranks during the period of our study. There is a chance of winning the market by constantly improving an app, such as adding new features and fixing bugs. However, developers should be careful in choosing the area in which they intend to succeed.

RQ3) What variables have a significant relationship with the ranks over time?

Motivation. In **RQ2**, we identify the variables that have a significant relationship with the rank trends. However, an app can face various rises and falls over time. We break the timeline of our subject apps with respect to the release dates of each app. Therefore, we can identify the variables that have a significant relationship with the ups and downs of the ranks over time. In this research question, we study the ranks with a finer grain (version), while, in the previous research question, we study the ranks with a broader view (i.e., rank trends).

Table 5: List of correlated variables.

Selected Variable	Correlated Variables
#Apps (Category)	#Paid Apps (Category)
#User-Reviews	#Negative, Positive, and Neutral User-Reviews, #User-Reviews with Negative, Positive, and Neutral Sentiment Scores
Address User-Reviews (Release Note)	Address User-Reviews (Description)
#Releases	Average Time Between Releases
#Downloads	Average #Downloads (Company)
Description Similarity	#Words in Description
Average Price (Company)	Price, #Paid Apps (Company)
Average Rating (Company)	#Apps (Company)

Approach. First, we remove the correlated variables with Spearman's $|\rho| > 0.7$ [66]. Table 5 shows the list of the correlated variables. Then, we build a mixed effects model [68, 101] to determine the variables that have a significant relationship with the ranks. A mixed effects model contains both fixed effect variables and random effect variables [27, 101]. A fixed effects variable is treated with a constant coefficient and intercept for all the observations, while the coefficient and the intercept of a random effect variable can vary across individual observations [68]. By applying a mixed effects model, we can explain the relationships between a dependent variable and the independent variables that are grouped according to one or more grouping factor(s). The mixed effects model can assume different intercepts or coefficients for each group [101].

Equation 2 shows the mixed effects model formula [68]. In Equation 2, g shows the grouping factor, n is the total number of variables, and m is the number of variables with fixed coefficients. Y_g holds the dependent variable. β_0 is the constant intercept and θ_{0g} is the intercept that varies across each grouping factor [68]. Let X_i denote the i^{th} independent variable; $\beta_i + \theta_{1g}$ and β_i indicate the coefficients of the X_i where θ_{1g} can vary across groups. ϵ_g is the indicator of errors. By having $\theta_{1g} = 0$, only the intercepts can vary. By setting $\theta_{0g} = 0$, only the coefficients can vary. By having both $\theta_{0g} = 0$ and $\theta_{1g} = 0$, the model would become a fixed effects model.

$$Y_g = \beta_0 + \theta_{0g} + \sum_{i=1}^{m} \beta_i X_i + \sum_{i=m}^{n} (\beta_i + \theta_{1g}) X_i + \epsilon_g \qquad (2)$$

We let the dependent variable to be the rank of each version of the apps. We set the independent variables to have random intercepts while keeping the coefficient of the independent variables fixed. Thus, we give a chance to the mixed effect model to tune variable intercepts according to the grouping factors but keep the coefficients union across different observations. Therefore, independent variables can contribute equally to the observations with a minor tune with respect to the grouping factors.

We group the independent variables according to (i) 13 rank trends and (ii) 30 areas nested within various versions of each individual app. Setting a grouping factor for rank trends gives a fair comparison between the apps. Instead of comparing all the apps together, we compare an app with its own competitors. A good analogy is the different weight classes in wrestling competitions. By having different weight classes, competitors can be judged on their techniques rather than their weights. With a similar reasoning as above, we set another grouping factor to the area of the app. We let different versions have varying intercepts as there might be other factors for each version but cannot be considered in a non-controlled empirical study [68], such as contextual requirements of the time in which a version is released.

Table 6: Mixed effects model of ranks, sorted by p − value.

Variable	χ^2	Pr(> |F|)		Estimate	Effect
Sentiment Score	62.09	3.55E-15	***	1.14E-01±1.45E-02	↗
#Sentences (Description)	46.10	1.19E-11	***	-5.08E-02±7.48E-03	↗
Name Similarity	41.16	1.47E-10	***	5.18E-02±8.08E-03	↗
Sentiment Score (Total)	23.19	1.49E-06	***	2.76E-02±5.73E-03	↗
Release Latency	19.71	9.13E-06	***	3.01E-02±6.78E-03	↘
Installation File Size	13.49	2.41E-04	***	1.79E-02±4.88E-03	↘
Star-Rating	12.07	5.15E-04	***	-4.20E-02±1.21E-02	↗
Appearance of New Topics (Release Note)	11.20	8.20E-04	***	2.75E-02±8.20E-03	↗
#Downloads	10.64	1.11E-03	**	-2.88E-02±8.82E-03	↗
Average Star-Rating (Category)	9.08	2.60E-03	**	-1.93E-02±6.42E-03	↘
Address User-Reviews (Release Note)	3.94	4.71E-02	*	-1.89E-02±9.51E-03	↗
Description Similarity	2.75	9.76E-02	.	2.98E-02±1.80E-02	↗
Average Price (Category)	2.50	1.14E-01		-1.31E-02±8.28E-03	↗
Appearance of New Topics (Description)	2.17	1.41E-01		1.23E-02±8.37E-03	↗
#Releases	1.67	1.97E-01		7.65E-03±5.93E-03	↗
Launch Date	1.39	2.38E-01		-1.91E-02±1.62E-02	↗
#Pictures	1.29	2.56E-01		-5.77E-03±5.08E-03	↗
Name Size	1.22	2.70E-01		-5.34E-03±4.84E-03	↘
Ratio of Paid Apps per Total (Category)	0.70	4.02E-01		-5.35E-03±6.38E-03	↗
#Apps (Category)	0.40	5.28E-01		-3.54E-03±5.60E-03	↘
Average Star-Rating (Company)	0.34	5.62E-01		3.19E-03±5.49E-03	↗
Deletion of Previous Topics	0.09	7.70E-01		2.28E-03±7.80E-03	↘
Average Price (Company)	0.08	7.75E-01		-6.41E-03±2.25E-02	↘
#User-Reviews	0.04	8.49E-01		-1.18E-02±6.22E-02	↗

$p - value$ codes: '***' < 0, '**' < 0.001, '*' < 0.01, '.' < 0.05

To estimate the goodness of fit of a mixed effects model, two measures are used [64]: (i) marginal R^2 and (ii) conditional R^2. The marginal R^2 describes the proportion of the variance explained by the fixed effects variables, while the conditional R^2 describes the proportion of the variance explained by both fixed and random effects variables. The marginal R^2 of our model is 0.02, but the conditional R^2 is 0.68. The lower value of the marginal R^2 compared to the conditional R^2 denotes that the proportion of the variance explained by both fixed and random variables is considerably higher than the proportion of the variance explained by the fixed variables. Thus, modeling the random effects greatly improves the explanatory power of our model [68].

Findings. Table 6 shows the output of the mixed effects model. Significant variables are highlighted with asterisks along with their effects on the ranks.

Observation 3_a**: User-reviews are important artifacts to explain the ranks.** The sentiment scores have a significant relationship with the ranks. Developers should take care of user-reviews and keep the users happy to achieve better ranks. For example, developers can resolve the bugs that are reported in the user-reviews. For extracting bug reports, developers can refer to the related work on this matter, such as Chen *et al.* [17], Villarroel *et al.* [96], and Sorbo *et al.* [25].

Observation 3_b**: Apps should be assigned with a proper name and description.** Name and description of an app should be related to the area of the app. Although there exist some apps, such as *Facebook*, whose own commercial name may not reflect their true

area, less famous apps should either find a way to advertise their apps or keep the similarity of names in mind.

Observation 3_c**: Developers should not wait too long to publish an update.** As the latency of an update increases for an app, the app tends to lose ranks. Such behavior might be caused by a probable Google's ranking strategy as Henze *et al.* [40] observed that releasing new updates is an effective strategy for increasing apps in App Store. Another possible explanation can be the users' preferences on seeing an updated version of an app earlier. Future studies should look into this matter more in-depth. Also, it is better to keep the installation files as small as possible.

Observation 3_d**: Introducing new topics, such as new features, helps developers to improve the ranks.** We observe that the introduction of new topics in the release notes has a significant positive relationship with the ranks. Developers can refer to the related work, such as Villarroel *et al.* [96], and Sorbo *et al.* [25], to mine the user-reviews and get some idea on the topics that they may want to add to the next versions.

Observation 3_e**: The common-wisdom does not always hold when it comes to improving the ranks.** Various variables, such as the number of pictures and the number of apps in a given category, may sound critical to have a better rank. For example, Tian *et al.* [90] observed that the number of pictures is an influential variable for receiving higher star-ratings. Conversely, as shown in Table 6, the number of pictures does not have a significant relationship with the ranks. In fact, only 11 variables share a significant relationship with the ranks.

> *App developers should take care of user-reviews and constantly improve the apps (e.g., introducing new features) to achieve higher ranks. Developers should be careful not to spend too much budget and time on the variables that sound important but actually do not share a significant relationship with the changes in the ranks.*

RQ4) How do app developers evaluate the findings?

Motivation. To better understand the practical value of our findings, we conducted a survey and in-depth interviews with app developers discussing our findings.

Approach. In this section, we describe our participants, the design of the survey, and the procedure of the survey.

Participants. The participants of our survey are mobile app developers. To find the participants, we contacted four app development companies that agreed to cooperate with us in conducting the survey. Finally, 51 app developers participated in our survey. We also had the opportunity to discuss our observations in detail with two developers from two different companies: one with over four years of app development experience and the other one with over five years.

Design. We gathered the developers' opinions using questionnaires. We asked five questions regarding the usefulness of our findings. The questions are listed in Table 7. Also, for each question, we asked the developers to provide further comments in case they did not agree with our findings or in case they would like to provide additional feedback (follow-up comments). We asked the developers to answer the questions on a five-point Likert scale: (i) strongly agree, (ii) agree, (iii) neutral, (iv) disagree, and (v) strongly disagree [50]. Table 8 shows the median and the distribution of the responses.

Table 7: List of the survey questions.

#	Question
1	Do you think that the studied areas are the top areas of mobile apps?
2	Are the considered variables reasonable and sufficient for explaining ranks and rank trends?
3	Do you think the findings of RQ1 are useful and practical for the industry?
4	Do you think the findings of RQ2 are useful and practical for the industry?
5	Do you think the findings of RQ3 are useful and practical for the industry?

Procedure. For each company, we presented our research and our results to the developers. It took around 30 minutes to do each presentation using slides. There was no time limit for developers to finish the survey. However, it took less than 15 minutes to have the questionnaires filled out by all the developers. Regarding the in-depth discussions, the first author conducted the interviews. Each interview took around 60 minutes.

Findings. The majority of the developers agree with our findings. For each question, first, we present the result of the survey. Then, we summarize our in-depth discussion with two developers.

(i) ***Do you think that the studied areas are the top areas of mobile apps?***

Survey. As shown in Table 8, developers agree that the investigated areas are the hot areas of mobiles apps. On the other hand, 11.8% and 2.0% of the developers disagree and strongly disagree with the question, respectively. According to the follow-up comments, one developer thinks that the area of *photography* might be important too. Two developers think that *social networking* is missing. However, as we look for the top keywords that are searched by real users, the exact name of the dominant social networking apps, such as *Facebook*, are more likely to be searched rather than searching for other alternative apps.

In-depth Discussions. Both developers approved the areas subject of our study with *agree* and *strongly agree*. One developer said *"you have covered all the top areas, such as dating, coupons, weather apps, health, and messaging"*.

(ii) ***Are the considered variables reasonable and sufficient for explaining ranks and rank trends?***

Survey. As shown in Table 8, 47.1% and 31.4% of the developers strongly agree and agree with the considered variables, respectively. However, 11.8% and 3.9% of the developers disagree with the variables. Summarizing the follow-up comments, developers that do not agree with our variables suggested the following variables: (i) code variables, (ii) team size, (iii) communication between the development team, (iv) expertness of developers, (v) marketing and advertisement, (vi) project plan, and (vii) the novelty of an app. Although we agree with the developers' suggestions, it is challenging to add more details about the code variables and team characteristics as such data is not publicly available. Future studies should look into the suggested variables in more details.

In-depth Discussions. Both developers confirmed our variables with *strongly agree*. They indicated that we have considered almost all the important variables.

(iii) ***Do you think the findings of RQ1 are useful and practical for the industry?***

Table 8: Median of the responses to each question the distribution of the responses. The number of the responses received for each item is reported in the parentheses.

#	Median	Distribution of the Responses				
		Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1	Agree	37.3% (19/51)	31.4% (16/51)	17.6% (9/51)	11.8% (6/51)	2.0% (1/51)
2	Agree	47.1% (24/51)	19.6% (10/51)	9.8% (5/51)	19.6% (10/51)	3.9% (2/51)
3	Strongly Agree	60.8% (31/51)	23.5% (12/51)	9.8% (5/51)	5.9% (3/51)	0.0% (0/51)
4	Strongly Agree	84.3% (43/51)	9.8% (5/51)	2.0% (1/51)	3.9% (2/51)	0.0% (0/51)
5	Strongly Agree	80.4% (41/51)	9.8% (5/51)	5.9% (3/51)	3.9% (2/51)	0.0% (0/51)

Survey. As shown in Table 8, 60.8% of the developer strongly agree with the findings of the first research question. On the other hand, according to the follow-up comments, one developer found the drops from the ranks very challenging for the developers of existing apps. This is due to our observation that most of the apps tend to fall in the ranks.

In-depth Discussions. Both developers confirmed the question with *strongly agree*. One developer mentioned: *"I think the reason that most apps have fallen in the ranking is the introduction of new apps to the market"*. Regarding competitiveness of different areas, he said: *"As an example, in the area of chess, new apps do not have serious barrier entry and can easily take the position of older apps by introducing new features"*. The other developer mentioned that seeing different rank trends in different areas is very interesting. It shows that choosing an area is an important decision.

(iv) *Do you think the findings of RQ2 are useful and practical for the industry?*

Survey. 84.3% of the developers strongly agree that the findings can be useful in practice.

In-depth Discussions. Both developers *strongly agreed* with the question. They said: *"We do not launch an app unless we make sure that our app can receive high star-ratings. Releasing newer versions normally include adding new features, fixing bugs, and improvement in the app, that all improve users' trust in our app. I think this is why the number of releases is an important variable. I also think there are a variety of apps that are better than the older ones, so the launch date makes much sense to me"* and *"I agree that the developers' tasks, such as constantly working on an app, introducing new topics, and releasing an updated version, lead to better ranks"*.

(v) *Do you think the findings of RQ3 are useful and practical for the industry?*

Survey. As shown in Table 8, 80.4% of the developers strongly agree with the findings. However, one developer thinks some suggestions, such as addressing user-reviews, may need a lot of effort.

In-depth Discussions. Both developers *strongly agreed* with the question. They said: *"Even not all the famous app names are irrelevant. For example, the names of famous apps, such as Facebook and Whatsapp, are quite relevant to their area"* and *"The highlighted factors, including the sentiment scores and installation file size, correctly show that the users need to be satisfied. Moreover, the release latency is an important issue. Developers need to address the issues quickly and release a newer version"*.

> *The majority of the developers agree that our findings can be very useful in practice.*

4 IMPLICATIONS FOR NEW APP DEVELOPERS

Newcomers to the app production rally, including start-up founders, entrepreneurs [5], and even a small team of developers, may be worried about the competitive nature of app markets [33, 88]. An important message of our paper for newcomers is that *"app developers should not stop improving their apps as they have a great chance to win the rally!"* (see observations 2_a and 2_b). Unfortunately, even if developers start with a novel idea, they are in danger of losing their ranks by other companies as soon as they reveal their idea (see observation 1_b). In other words, newcomers need to have a strong personal drive to success [16]. It is not surprising that the lack of discipline and work could make it infeasible to succeed [6].

The key points for newcomers that are discussed in this paper can show them the right path.

(i) Developers should carefully select the area in which they intend to work. Developers should do their homework and see if there is a huge number of dominant apps in an area or not. The most wanted areas of apps can be a decent start (see observations 1_c and 2_d).
(ii) *Never is late* as in the past two years, many apps (other than our subject apps) have achieved higher ranks in Google Play Store (see observation 2_a).
(iii) Investing time and money on the user-reviews, adding new features, and releasing improved versions can give developers a higher chance to win the rally (see observations 2_b, 3_a, 3_c, and 3_d).
(iv) Developer should work on the variables that can potentially improve the rank of an app. Developers should not be distracted by the variables that sound critical but are not much important (see observation 3_e).

Developers of new apps with no (or a limited number of) user-reviews can study the user-reviews, features, and descriptions of other apps to get some hints. We investigate the descriptions and release notes of each app in each area and compare them with the user-reviews of other apps in the same area. We notice that, on average, there is a similarity of 63% between the descriptions and release notes of each app and the user-reviews of other apps. Newcomers can identify popular and repelling features by studying other apps within the same area.

5 THREATS TO VALIDITY

In this section, we discuss the potential threats to the validity of our experiments [87].

5.1 Construct Validity

Martin *et al.* [57] reported that using an incomplete set of user-reviews in Blackberry World app store can introduce bias to the findings. To mitigate such a threat, we gradually retrieved all the user-reviews of our subject apps for two years. Moreover, Google Play Store does not reveal the information about all the available apps. We mitigate this limitation by clicking on *"Show More"* button as many times as possible to retrieve all the accessible apps.

One way to capture app features is decompiling installation files to byte-codes. Unfortunately, Google Play Store only reveals the

latest version of the apps. We captured the concepts of features by breaking down the release notes and descriptions as Johann *et al.* [45] reported a precision of up to 88% for app descriptions in identifying app features.

5.2 External Validity

We rely on Google to get the app rankings. Our findings and experiments will be still useful even if Google completely changes the ranking algorithm in the future for three main reasons. First, the majority of our variables are related to users and developers activities, such as addressing user-reviews and introducing new topics. Second, the same approach can be used to find the most important variables in the future with the most current data. Third, Google improves the ranking algorithm over time [77]. However, the variables that are identified in this paper have constantly appeared as the variables that share a significant relationship with the ranks for two years.

We retrieve the apps that can be accessed via Google Play Store. Fortunately, Google gives a chance to all apps to appear in the rankings as we published a sample app in Google Play Store with no download and no star-rating. Regardless, as our app was published recently in Google Play Store, we could have it on our radar of ranks. Therefore, we believe that our set of apps is a combination of all sort of apps. In addition, the top areas may be subject to change in different geographical locations and time. However, the emphasis of our work is on the differences in areas.

If a company is not interested in the ranks, the result of our study may not be generalized to the usage of such a company. However, it would be still a great opportunity for such a company to improve its rank and attract more customers.

5.3 Internal Validity

We study 45 variables to explain the ranks. However, we do not claim an exhaustive list of explanatory variables. Future work can shed more light on explaining the ranks by taking more variables into consideration. Some variables may not directly impact the ranks. For example, adding a new feature to an app may make its users more satisfied. As a result, the rank of the app can be improved.

6 RELATED WORK

In this section, we summarize the related work along two research directions: (i) empirical studies and (ii) user-review analyses.

6.1 Empirical Studies

Several studies have been conducted to identify the variables that share significant relationships with the number of downloads or star-ratings [58]. Kim *et al.* [46] studied star-ratings of mobile apps. They showed that the star-ratings could affect the users' decision in downloading an app. Linares-Vasquez *et al.* [52] showed that the quality of the APIs affects the star-ratings of the apps. Bavota *et al.* [10] indicated the high correlation between the star-ratings of the apps and the fault- and change-proneness of the APIs used by apps. Noei *et al.* [68] observe that not only the app variables can impact the star-ratings, but also some device variables, such as display size, have a strong relationship with star-ratings. Lee *et al.* [48] mined 300 free and paid iOS apps. They observed a positive relationship between the download ranks and app page contents. Henze and Boll [40] studied the relationship between the release time and user activities in App Store. They observe that releasing new updates can improve the ranks in App Store. Coelho *et al.* [20] mined app issues to investigate the exceptions that happen in apps. Fu *et al.* [28] analyzed star-ratings and user-reviews by applying topic modeling techniques. Linares-Vasquez *et al.* [53] noticed that anti-patterns in source code negatively affect app quality. Ruiz *et al.* [84] recommend developers not to release incomplete apps too early to avoid receiving low star-ratings. None of the earlier work has clustered and investigated the app ranks. Collecting a large data and considering each release of apps in two years allows us to provide practical guidelines for improving a rank.

6.2 User-Review Analyses

Many researchers aim to study the user-reviews and give developers some insight regarding users' demands and experience [36, 73, 75, 96]. Ciurumelea *et al.* [19] organized user-reviews concerning the users' requests. They recommend source code to developers using code localization. Palomba *et al.* [73] propose a solution to recommend the source code changes that are needed to address the user-reviews by measuring the asymmetric Dice similarity coefficient [7] between the words in user-reviews and source code. Villarroel *et al.* [96], Sorbo *et al.* [25], Panichella *et al* [75], Chen *et al.* [17], and Gu and Kim [36] aim to cluster the user-reviews into meaningful groups. For instance, Villarroel *et al.* [96] provided a solution to classify the user-reviews into groups of bug reports and feature requests. The above solutions can help developers to get the most out of the user-reviews of their own apps. Developers can use such solutions along with our findings.

7 CONCLUSION

In this paper, we study the evolution of ranks in Google Play Store. We observe that there are 13 trends in the ranks, including falling, maintaining, and rising trends. In some areas, such as *budget*, existing apps tend to maintain their ranks. Therefore, it is harder for newer apps to compete with the existing apps. However, by drawing a regression model, we observe that app developers should not be afraid of a late entry into the market since the apps that appeared later achieved higher ranks during our study. There is a chance of improving the ranks by continuously improving the app, such as adding new features and fixing bugs. We also model the ranks across apps and versions of each app. We identified 11 variables that share a significant relationship with the changes in the ranks, such as the name and the appearance of new topics. Furthermore, developers should not be distracted by some variables that sound important but do not share a significant relationship with the ranks, such as the number of pictures. Finally, the feedback obtained from 51 app developers confirms that our findings can help app developers to achieve higher ranks in Google Play Store.

Future work should study the ranks in other app markets, such as Apple App Store. The differences and the similarities of various app markets can give insights to cross-platform app developers.

REFERENCES

[1] Bram Adams and Shane McIntosh. 2016. Modern release engineering in a nutshell–why researchers should care. In *Proceedings of 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 5. IEEE, 78–90.

[2] Charu C Aggarwal and ChengXiang Zhai. 2012. *Mining text data*. Springer Science & Business Media.

[3] John H Aldrich and Forrest D Nelson. 1984. *Linear probability, logit, and probit models*. Vol. 45. Sage.

[4] Allacronyms. 2017. All Acronyms. [Online] Available: https://www.allacronyms.com/.

[5] David B Audretsch and Max Keilbach. 2004. Entrepreneurship and regional growth: an evolutionary interpretation. *Journal of Evolutionary Economics* 14, 5 (2004), 605–616.

[6] Bill Aulet. 2013. *Disciplined entrepreneurship: 24 steps to a successful startup*. John Wiley & Sons.

[7] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.

[8] Normi Sham Awang Abu Bakar and Iqram Mahmud. 2013. Empirical analysis of android apps permissions. In *Proceedings of the 2013 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*. IEEE, 406–411.

[9] Victor R Basili. 1992. *Software modeling and measurement: the Goal/Question/Metric paradigm*. Technical Report. University of Maryland at College Park.

[10] Gabriele Bavota, Mario Linares-Vasquez, Carlos Eduardo Bernal-Cardenas, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. 2015. The Impact of API Change-and Fault-Proneness on the User Ratings of Android Apps. *IEEE Transactions on Software Engineering* 41, 4 (2015), 384–407.

[11] Steven Bellman, Robert F Potter, Shiree Treleaven-Hassard, Jennifer A Robinson, and Duane Varan. 2011. The effectiveness of branded mobile phone apps. *Journal of interactive Marketing* 25, 4 (2011), 191–200.

[12] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series.. In *KDD workshop*, Vol. 10. Seattle, WA, 359–370.

[13] James C Bezdek, Robert Ehrlich, and William Full. 1984. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences* 10, 2-3 (1984), 191–203.

[14] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research* 3 (2003), 993–1022.

[15] Juan Cao, Tian Xia, Jintao Li, Yongdong Zhang, and Sheng Tang. 2009. A density-based method for adaptive LDA model selection. *Neurocomputing* 72, 7 (2009), 1775–1781.

[16] Gary J Castrogiovanni. 1996. Pre-startup planning and the survival of new small businesses: Theoretical linkages. *Journal of management* 22, 6 (1996), 801–822.

[17] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 767–778.

[18] Ronald Christensen. 2006. *Log-linear models and logistic regression*. Springer Science & Business Media.

[19] Adelina Ciurumelea, Andreas Schaufelbhl, Sebastiano Panichella, and Harald Gall. 2017. Analyzing Reviews and Code of Mobile Apps for Better Release Planning. In *Proceedings of the 24th International Conference on Software Analysis Evolution and Reengineering*. IEEE.

[20] Roberta Coelho, Lucas Almeida, Georgios Gousios, and Arie van Deursen. 2015. Unveiling exception handling bug hazards in Android based on GitHub and Google code issues. In *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 134–145.

[21] Enrique Costa-Montenegro, Ana Belen Barragans-Martinez, and Marta Rey-Lopez. 2012. Which App? A recommender system of applications in markets: Implementation of the service for monitoring users' interaction. *Expert systems with applications* 39, 10 (2012), 9367–9375.

[22] Diya Datta and Sangaralingam Kajanan. 2013. Do app launch times impact their subsequent commercial success? an analytical approach. In *Proceedings of the 2013 International Conference on Cloud Computing and Big Data (CloudCom-Asia)*. IEEE, 205–210.

[23] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, Vol. 6. 449–454.

[24] Romain Deveaud, Eric SanJuan, and Patrice Bellot. 2014. Accurate and effective latent concept modeling for ad hoc information retrieval. *Document numérique* 17, 1 (2014), 61–84.

[25] Andrea Di Sorbo, Sebastiano Panichella, Carol V Alexandru, Junji Shimagaki, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. 2016. What would users change in my app? summarizing app reviews for recommending software changes. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 499–510.

[26] Thomas A Domencich and Daniel McFadden. 1975. *Urban travel demand-a behavioral analysis*. Technical Report.

[27] Julian J Faraway. 2005. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press.

[28] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. 2013. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*. ACM, 1276–1284.

[29] Laura V Galvis Carreño and Kristina Winbladh. 2013. Analysis of user comments: an approach for software requirements evolution. In *Proceedings of the 35th International Conference on Software Engineering*. IEEE, 582–591.

[30] Rajiv Garg and Rahul Telang. 2012. Inferring app demand from publicly available data. (2012).

[31] Google. 2017. Android Developers Blog. [Online] Available: https://android-developers.googleblog.com/2017/02/welcome-to-google-developer-day-at-game.html/.

[32] Google. 2017. Get discovered on Google Play search. [Online] Available: https://support.google.com/googleplay/android-developer/answer/4448378/.

[33] Google. 2017. Google play store. [Online] Available: http://play.google.com/.

[34] Tom Griffiths. 2002. Gibbs sampling in the generative model of latent dirichlet allocation. (2002).

[35] Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *the National Academy of Sciences* 101, suppl 1 (2004), 5228–5235.

[36] Xiaodong Gu and Sunghun Kim. 2015. " What Parts of Your Apps are Loved by Users?"(T). In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 760–770.

[37] Mark Harman, Yue Jia, and Yuanyuan Zhang. 2012. App store mining and analysis: MSR for app stores. In *Proceedings of the 9th Working Conference on Mining Software Repositories*. IEEE, 108–111.

[38] Harrell. 2017. Harrell. [Online] Available: http://cran.r-project.org/web/packages/Hmisc/index.html.

[39] Frank E. Harrell. 2001. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer.

[40] Niels Henze and Susanne Boll. 2011. Release your app on sunday eve: Finding the best time to deploy apps. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 581–586.

[41] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*. Vol. 398. John Wiley & Sons.

[42] Claudia Iacob and Rachel Harrison. 2013. Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*. IEEE, 41–44.

[43] Md Rakibul Islam and Minhaz F Zibran. 2017. Leveraging automated sentiment analysis in software engineering. In *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, 203–214.

[44] Jazzy. 2017. Jazzy Spell Checker. [Online] Available: http://jazzy.sourceforge.net/.

[45] Timo Johann, Christoph Stanik, Walid Maalej, et al. 2017. Safe: A simple approach for feature extraction from app descriptions and app reviews. In *Proceedings of the 25th International Conference on Requirements Engineering*. IEEE, 21–30.

[46] Hee-Woong Kim, HL Lee, and JE Son. 2011. An exploratory study on the determinants of smartphone app purchase. In *Proceedings of the 11th International DSI and the 16th APDSI Joint Meeting*.

[47] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on empirical methods in natural language processing*. Association for Computational Linguistics, 423–430.

[48] Gunwoong Lee and TS Raghu. 2011. Product Portfolio and Mobile Apps Success: Evidence from App Store Market.. In *AMCIS*.

[49] T Warren Liao. 2005. Clustering of time series data survey. *Pattern recognition* 38, 11 (2005), 1857–1874.

[50] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

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

[52] Mario Linares-Vásquez, Gabriele Bavota, Carlos Bernal-Cárdenas, Massimiliano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. 2013. API Change and Fault Proneness: A Threat to the Success of Android Apps. In *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering*. ACM, 477–487.

[53] Mario Linares-Vásquez, Sam Klock, Collin McMillan, Aminata Sabané, Denys Poshyvanyk, and Yann-Gaël Guéhéneuc. 2014. Domain matters: bringing further evidence of the relationships among anti-patterns, application domains, and quality-related metrics in java mobile apps. In *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 232–243.

[54] Julie B Lovins. 1968. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory.

[55] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA., 281–297.

[56] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60.

[57] William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang. 2015. The app sampling problem for app store mining. In *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE, 123–133.

[58] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. 2016. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering* (2016).

[59] James T Sincich McClave, Terry James T McClave, and Terry Sincich. 2006. *Statistics*. Number QA 276.12. M33 2006.

[60] Daniel McFadden et al. 1973. Conditional logit analysis of qualitative choice behavior. (1973).

[61] Stuart McIlroy, Nasir Ali, and Ahmed E Hassan. 2016. Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store. *Empirical Software Engineering* 21, 3 (2016), 1346–1370.

[62] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[63] Glenn W Milligan and Martha C Cooper. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* 50, 2 (1985), 159–179.

[64] Shinichi Nakagawa and Holger Schielzeth. 2013. A general and simple method for obtaining R2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution* 4, 2 (2013), 133–142.

[65] Netlingo. 2017. Top 50 Most Popular Text Terms. [Online] Available: http://www.netlingo.com/top50/popular-text-terms.php.

[66] Thanh HD Nguyen, Bram Adams, and Ahmed E Hassan. 2010. Studying the impact of dependency network measures on software quality. In *Proceedings of the 26th International Conference on Software Maintenance*. IEEE, 1–10.

[67] Ehsan Noei and Abbas Heydarnoori. 2016. EXAF: A search engine for sample applications of object-oriented framework-provided concepts. *Information and Software Technology* 75 (2016), 135–147.

[68] Ehsan Noei, Mark D Syer, Ying Zou, Ahmed E Hassan, and Iman Keivanloo. 2017. A study of the relation of mobile device attributes with the user-perceived quality of android apps. *Empirical Software Engineering* (2017), 1–29.

[69] Christiane Nord. 2005. *Text analysis in translation: Theory, methodology, and didactic application of a model for translation-oriented text analysis*. Number 94. Rodopi.

[70] Optimaize. 2017. Language Detection Library for Java. [Online] Available: https://github.com/optimaize/language-detector/.

[71] Dennis Pagano and Walid Maalej. 2013. User feedback in the appstore: An empirical study. In *Proceedings of the 21st International Conference on Requirements Engineering*. IEEE, 125–134.

[72] Fabio Palomba, Mario Linares-Vásquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. 2015. User Reviews Matter! Tracking Crowdsourced Reviews to Support Evolution of Successful Apps. In *Proceedings of the 31st International Conference on Software Maintenance and Evolution*. IEEE, 291–300.

[73] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. 2017. Recommending and localizing change requests for mobile apps based on user reviews. In *Proceedings of the 39th International Conference on Software Engineering*. 106–117.

[74] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, C Visaggio, Gerardo Canfora, and H Gall. 2015. How can i improve my app? classifying user reviews for software maintenance and evolution. In *Proceedings of the 31st International Conference on Software Maintenance and Evolution*.

[75] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. 2016. ARdoc: app reviews development oriented classifier. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 1023–1027.

[76] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*. Association for Computational Linguistics, 38–41.

[77] Sarah Perez. 2017. Google Play now considers user engagement, not just downloads, in ranking games. [Online] Available: https://beta.techcrunch.com/2017/02/28/google-play-now-considers-user-engagement-not-just-downloads-in-ranking-games/.

[78] Jose Pinheiro, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, et al. 2007. Linear and nonlinear mixed effects models. *R package version* 3 (2007), 57.

[79] Quora. 2017. What are the downsides to releasing frequent app updates? [Online] Available: https://www.quora.com/What-are-the-downsides-to-releasing-frequent-app-updates/.

[80] Adrian E Raftery, Steven Lewis, et al. 1992. How many iterations in the Gibbs sampler. *Bayesian statistics* 4, 2 (1992), 763–773.

[81] Anand Rajaraman, Jeffrey D Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. 2012. *Mining of massive datasets*. Vol. 77. Cambridge University Press Cambridge.

[82] Alan A Ramaley, Darrell L Aldrich, David M Buchthal, and Thomas W Olsen. 2003. Method for managing embedded files for a document saved in HTML format. US Patent 6,585,777.

[83] Eric Ries. 2011. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books.

[84] Israel J Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Thorsten Berger, Steffen Dienst, and Ahmed E Hassan. 2017. Examining the Rating System Used in Mobile-App Stores. *IEEE Software* 33, 6 (2017), 86–92.

[85] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[86] Semrush. 2017. Semrush. [Online] Available: https://www.semrush.com/.

[87] Forrest Shull, Janice Singer, and Dag I.K. Sjøberg. 2007. *Guide to Advanced Empirical Software Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[88] Statista. 2017. Number of apps available in leading app stores as of 2017. [Online] Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores.

[89] Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E Hassan, and Kenichi Matsumoto. 2017. An empirical comparison of model validation techniques for defect prediction models. *IEEE Transactions on Software Engineering* 43, 1 (2017), 1–18.

[90] Yuan Tian, Meiyappan Nagappan, David Lo, and Ahmed E Hassan. 2015. What are the characteristics of high-rated apps? a case study on free android applications. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 301–310.

[91] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 2 (2001), 411–423.

[92] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H Dieter Rombach. 2002. Goal question metric (gqm) approach. *Encyclopedia of software engineering* (2002).

[93] Brandon K Vaughn. 2008. Data analysis using regression and multilevel/hierarchical models, by Gelman, A., & Hill, J. *Journal of Educational Measurement* 45, 1 (2008), 94–97.

[94] Pradeep K Venkatesh, Shaohua Wang, Feng Zhang, Ying Zou, and Ahmed E Hassan. 2016. What Do Client Developers Concern When Using Web APIs? An Empirical Study on Developer Forums and Stack Overflow. In *Proceedings of the 2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 131–138.

[95] Luigi Vigneri, Jaideep Chandrashekar, Ioannis Pefkianakis, and Olivier Heen. 2015. Taming the android appstore: lightweight characterization of android applications. *CoRR, abs/1504.06093* (2015).

[96] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 14–24.

[97] Sanford Weisberg. 2005. *Applied linear regression*. Vol. 528. John Wiley & Sons.

[98] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[99] Lotfi Asker Zadeh. 1988. Fuzzy logic. *Computer* 21, 4 (1988), 83–93.

[100] Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen. 2015. Discovery of ranking fraud for mobile apps. *IEEE Transactions on knowledge and data engineering* 27, 1 (2015), 74–87.

[101] Alain Zuur, Elena N Ieno, Neil Walker, Anatoly A Saveliev, and Graham M Smith. 2009. *Mixed effects models and extensions in ecology with R*. Springer Science & Business Media.