Hall-of-Apps: The Top Android Apps Metadata Archive

Laura Bello-Jiménez, Camilo Escobar-Velásquez, Anamaria Mojica-Hanke,
Santiago Cortés-Fernández, Mario Linares-Vásquez
Systems and Computing Engineering, Universidad de los Andes, Bogotá, Colombia
{ln.bello10,ca.escobar234,ai.mojica10,s.cortes,m.linaresv}@uniandes.edu.co

ABSTRACT

The amount of Android apps available for download is constantly increasing, exerting a continuous pressure on developers to publish outstanding apps. Google Play (GP) is the default distribution channel for Android apps, which provides mobile app users with metrics to identify and report apps quality such as rating, amount of downloads, previous users comments, etc. In addition to those metrics, GP presents a set of top charts that highlight the outstanding apps in different categories. Both metrics and top app charts help developers to identify whether their development decisions are well valued by the community. Therefore, app presence in these top charts is a valuable information when understanding the features of top-apps. In this paper we present **Hall-of-Apps**, a dataset containing top charts' apps metadata extracted (weekly) from GP, for 4 different countries, during 30 weeks. The data is presented as (i) raw HTML files, (ii) a MongoDB database with all the information contained in app's HTML files (*e.g.,* app description, category, general rating, etc.), and (iii) data visualizations built with the `D3.js` framework. A first characterization of the data along with the urls to retrieve it can be found in our online appendix: **https://bit.ly/2uIkXs8**. Dataset: **https://bit.ly/2Wu9qsn**

CCS CONCEPTS

• **Information systems → Web mining**; • **Human-centered computing → Empirical studies in ubiquitous and mobile computing**.

KEYWORDS

Google Play, Top Chart Apps, Android, Metadata

ACM Reference Format:
Laura Bello-Jiménez, Camilo Escobar-Velásquez, Anamaria Mojica-Hanke, Santiago Cortés-Fernández, Mario Linares-Vásquez. 2020. **Hall-of-Apps**: The Top Android Apps Metadata Archive. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387497

1 INTRODUCTION

Nowadays, there are almost three million apps in the Google Play Store (GP)[3]; this number is increasing every year as developers' interests are focused more on mobile development [1]. Nevertheless, success of mobile apps do not rely only on the uniqueness of the idea, but also on the capabilities of the app's owner team. Since GP uses ratings and comments as a way to measure acceptance of an app by the users, mobile app developers must be aware of the needs, recommendations, requests and errors that users report at GP. Because of this, if a new app is planned to be released, a comprehensive study of the behavior of similar apps through time should be done to ensure the success of the app. Although, current approaches for defining this release strategies are built using small samples of alike apps and in some cases they only use snapshots (*i.e.,* a period of time of the whole app story). Therefore, analyzing information extracted from incomplete apps histories, can lead to improper definitions of the factors that characterize the success of an application over time [25], [6].

To help developers and researchers when dealing with release engineering and app store analysis (for Android apps) based on poor datasets, we have extracted and processed the metadata of GP categories' top charts for 4 different countries along 30 weeks. This information is publicly available in the **Hall-of-Apps** [7]dataset. With **Hall-of-Apps**, we aim to provide practitioners and researchers with a dataset that takes into account changes over time of apps published in the GP. We expect **Hall-of-Apps** will enable the study of trends and features of the apps that survive in GP top charts after several weeks, in different countries. Additionally, it could enrich the set of available resources for developers to generate solid release strategies based on particular characteristics that make apps successful. **Hall-of-Apps** includes metadata information (*e.g.,* description, price, category, collection), and user feedback-related information such as reviews and ratings.

Prior to the release of **Hall-of-Apps**, most of the existing datasets have focused on extracting metadata from GitHub and GP [9], finding rapid releases on GitHub projects [16] or discovering permissions changes [30]. However, to the best of our knowledge, there is no publicly available dataset that contains raw and weekly-indexed information about top apps stored in GP which takes into account metadata over different periods of time.

The rest of this paper is organized as follows: Section 2 describes **Hall-of-Apps** and preliminary statistics about the data; in particular, Section 2.1 presents the methodology used to extract and parse the data, and Section 2.2 details the used storage mechanism. Section 3 shows the faced limitations and challenges. Previous research and related datasets are presented in Section 4. On top of that, potential uses and improvements to the dataset are depicted in Section 5. Finally, our conclusions are presented in Section 6.

MSR '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387497

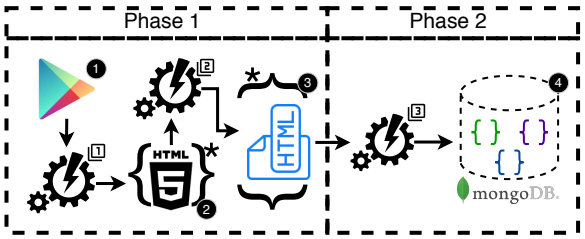

Figure 1: Temporal Extraction diagram.

2 THE DATASET

In this section we present **Hall-of-Apps** [7], a dataset containing top charts' apps metadata and reviews, extracted (weekly) from Google Play during 30 weeks, starting on November, 2017 until May, 2018. Each week we extracted the information of the top 100 apps available in the top charts of 33 Google Play categories and the Editor Choice list (ECL)[1]. It is important to notice that each GP category has two collections: (i) top free and (ii) top paid, therefore, we collected top 100 apps for 66 collections plus the editor choice list.

Additionally, in order to enhance the capabilities of our dataset, for each week we extracted the aforementioned information from four countries with different languages: Brazil, Colombia, Germany, and USA. The current release of **Hall-of-Apps** has the metadata of **more than 670K apps**[2] **(~22k per week) and about 19M reviews**.

2.1 Data extraction

Fig. 1 depicts the extraction process executed weekly to obtain the information stored in the dataset. The process consist of two phases: **Scrapping** and **HTML parsing**.

Google Play Scraping: This phase consist of extracting the HTML representation of each app belonging to the top charts (free and paid) of 33 categories and the ECL. To this, we created two NodeJS scripts: the first one in charge of extracting the list of apps in all the categories' top chart, and the second one in charge of going through the aforementioned list extracting the associated HTML file for each element of the list.

In order to build the first script, we had to understand how the GP URLs were created, therefore, we analyzed the URLs of the 67 top charts to identify existing patterns (❶ in Fig. 1); as a result we identified that GP had a predefined URL template capable of pointing to any category' top chart after performing small changes. The template is:

```
https://play.google.com/store/apps/category/<CA>/collection/<COL>?hl=en&gl=<CO>
```

, where <CA> is the GP app category (*e.g.,* ART_AND_DESIGN, BEAUTY, etc.), <COL> is the GP collection (*i.e.,* topselling_free, topselling_paid) and <CO> is the country (*i.e.,* co, br, de, us). Taking into account this, our first effort was to create the list of all possible URL permutations, this list has 268 URLs (67 top charts × 4 countries) that can be accessed in our online appendix[4]. With this in mind, we started requesting the content from the URLs (1 in Fig. 1). In a first experiment, we identified that GP only returns the first 10 apps if a GET request was made. Therefore, after studying the network requests done when refreshing the page, we identified that a POST request could be made with the following JSON as parameter:

```
{"start":0,"num":100,"numChildren":0,"ipf":1,"xhr":1}
```

The extra JSON parameter allowed us to retrieve the first 100 apps for the requested URL. Once we were able to request correctly the list of apps, our next step was parsing the request response to identify the URLs for each app in the list. We used the *cheerio* npm package [2] to parse and query the response's HTML with specific identificators/selectors associated to the apps. Afterwards, we created a list of all apps URLs for the second script to visit and extract the detailed information (❷ in Fig. 1). An example of an element of the aforementioned list is:

```
{
    "weekName": "20200202-20200208",
    "category": "education",
    "appName": "com.brain2016.speaking",
    "country": "co",
    "url":  "https://play.google.com/store/apps/details?id=com.brain2016.
          speaking&hl=en&gl=co"
}
```

The second script (2 in Fig. 1) reads the generated list and performs a request for each element; once the HTML is returned, it is stored in a folder (❸ in Fig. 1), using the following naming structure:

```
<weekInterval>%<CO>%<CA>%<appPackage>.html
```

It is important to notice that an app might appear more than once if it belongs to more than one category top chart. As a result of this first phase, we collected 671,041 HTML files for apps in the categories' top charts of four different countries. These HTML files are the input to our MongoDB dataset generation process.

HTML Parsing: In order to be able to process the HTML files, first we manually analyzed the tags structure of the files. Thus, we randomly selected a subset of the complete set of files and searched for the tags and text patterns to gather the information we required from the HTML snippets. In particular, this subset had files for different categories, countries, top charts and weeks to improve the representativeness of the subset over the complete set. After understanding the HTML files structure, by using the random sample, we developed an HTML parser written in Python (using the BeautifulSoup library) to search the targeted tags and extract its content, before being stored in a database (3 in Fig. 1). Finally, the HTML parser was used on the complete list of files to populate a NoSQL database (❹ in Fig. 1) with three collections: *app*, *review* and *extra app*.

During the parsing, we discovered that from weeks 17 to 30, the HTML format changed, thus, it was necessary to search for the new tags and adapt the parser to recognize when the file had the old or the new format. For the app metadata, the new format included different categorical values. Additionally, in the new format, reviews and similar apps information, were not stored in HTML tags but in JavaScript arrays, in consequence, it was imperative to recognize the position of each field inside the array and adapt the parser to extract this information. Besides, when performing this analysis, we found the reviews' date was written in milliseconds, thus, we didn't transform it to date because the time zone could be different

[1]The Editor Choice List is a selection of top notch apps and games handpicked by Google Play editors.
[2]Our definition of an app takes into account the category and week it was extracted.

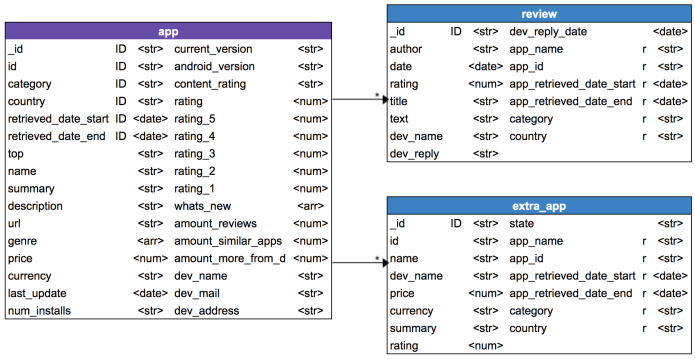

Figure 2: Non-relational database diagram.

from the original source. To incorporate the changes in the parser we ran the parsing process for a second time.

2.2 Data storage

Hall-of-Apps has two storage mechanisms. The first one is raw HTML files; these files are stored by week and grouped by month. Each file has a predefined naming structure to identify the week date in which the apps were extracted, app package, country, collection (*i.e.,* top free or top paid) and category. The second one, is a MongoDB database. We decided to use a NoSQL database due to the redundancy inherited in the extracted data, which in case of being stored in a relational database, it would require to follow the ACID properties[13], and this would generate several tables due to the nature of the data. Additionally, since the text fields in our dataset do not have an established length, if a SQL database were used it would generate a disc partition when using a relational schema.

The MongoDB database consists of three data collections, depicted in Fig. 2. The main collection is called *app*, because it keeps characteristic information about the app and retrieval dates; however, each app could have reviews, similar apps and more apps from the same developer, thus, because reviews and extra apps have large text fields, we created two new collections, *review* and *extra app*. With these two additional collections we wanted to avoid overloading the main collection.

On the one hand, the *review* collection has information about the author, rating, date and, if the developer wrote a response, it has the text and date. On the other hand, *extra app* has information about similar apps and others created by developer; to distinguish them, each app has a *state* field which indicates whether the app is a similar app or another app from the same developer.

Besides, since an app could appear in another country, in a different week or in a different top, we defined ids composed by: retrieved date start and end, app package, category and country.

2.3 Hall-of-Apps Description

When uncompressed, the **Hall-of-Apps** [7] HTML files weight 231GB, thus, the files are available — in our appendix — in a `.tar` archive and compressed with a compression algorithm called Burrows-Wheeler (BZ2) to generate a `.tar.bz2` with a smaller size of 38GB. Moreover, the MongoDB database was exported using `mongodump`, an utility for creating binaries of the contents, and compressed into a .zip file of size 3.2GB. Both, HTML files and dump, were uploaded to Zenodo to enable further studies. To download the dataset [7] and find further description of **Hall-of-Apps**, metrics, and tools, we encourage the reader to visit our online appendix [4].

Table 1: Number of HTML files in Hall-of-Apps.

Month	Year	# of extracted files
November	2017	87,700
December	2017	111,989
January	2018	90,154
February	2018	89,286
March	2018	89,545
April	2018	112,459
May	2018	89,908

D3 Visualizations: In order to facilitate the understanding of our dataset, we generated visualizations that aim to explain some metrics, distributions and statistics of the data in the NoSQL database. The visualizations are also available in our online appendix [4].

The scripts used to generate the visualizations are written in JavaScript using the Data Driven Documents library (`D3.js`). Each of the `D3.js` scripts contains a function that produces a particular chart/visualization (for example, a Stacked Bar Chart) using as input `CSV` files that contain synthesized information of various aggregations and/or queries made with the database.

Metrics: We have collected information over 30 weeks with a total of 671,041 files and apps, in the raw HTML files and database respectively. Table 1 outlines the amount of extracted files grouped by month. In addition, the *review* collection has 19,095,412 documents, and the *extra app* collection has 11,415,027 documents. Both collections have more records than *apps*, because each app can have between 0 to 40 reviews (as scrapped from GP), between 0 to 16 similar apps, and between 0 to 16 more apps from the same developer.

It is worth noting that, despite the fact we requested the first 100 apps in each top, GP not always returned that amount, because of two reasons: (i) sometimes the IP of the machine that was extracting the HTML was banned and, (ii) in not all the cases there were top 100 apps for a category. In consequence, the amount of data changes over time and we could discover some interesting behaviors, for example, GP has a predefined amount of apps in the editor choice depending the country. Detailed information and visualizations about data types and distribution are within our appendix [4].

3 LIMITATIONS, CHALLENGES

We ran into several difficulties, challenges and limitations that arose from internal and external threats. For example, HTML files don't have the same information for all apps, hence, we had to analyze different files from different categories and countries to keep track of all the attributes to extract. Besides, GP changed the format of HTML files in the 17th week; we overcome this challenge by adapting the HTML parser, after performing a new manual analysis of the web pages. We had to search for new tags and to extract reviews and extra apps from a JavaScript array contained inside the HTML. Other limitations are described as follows.

Banned IP: During the time we collected the information, we used the same virtual machine to request information from GP, that's why they banned our machine IP and impeded to pull out HTML files over 4 weeks in Colombia and 1 week in Brazil, in the ECL and in the top free and paid lists of some categories, respectively.

URLs: To download the HTML files, we discovered that GP had a particular system of URLs. Then, we only had to iterate over the URLs changing category, top, language and country to find apps in the top in different countries. Nevertheless, they switched that system, and, nowadays, our HTML scraper cannot extract information from the website following the same procedure.

Information loss: While we were downloading the HTML files, we didn't keep track of the position of the apps in the lists and the HTML itself doesn't include that information. Therefore, in our dataset we only have the set of apps that belonged to the top charts but without any order, *i.e.,* without the position in the lists.

4 RELATED WORK

When analyzing apps' metadata from the GP market, the research community has focused its efforts on different subfields [25] such as, API analysis, feature analysis, release engineering, reviews analysis, and security, among others. Most of the previous studies used metadata to extract features and bugs from reviews and description [8, 10, 12, 14, 15, 23], while others generate new GP categories using clustering and classification models [17, 19, 22, 26]. Other studies have investigated users behavior such as the installation trends of users with recently updated apps [18, 27] or studied the impact of releases on the number of downloads [21]. Additionally, release planning using user reviews analysis has been investigated [31]. However, the aforementioned works were based only on metadata from a single time snapshot, *i.e.,* the studies did not include metadata extracted over different periods of time, which could lead to partial/limited analyses of behaviors and results.

Few studies used metadata extracted over different periods of time; for instance, CIRA [24] is a tool that mined GP and Windows Phone Store over a 12 month time period to identify the releases with most impact on ratings and downloads. In addition, Lee and Raghu [20] extracted top apps' metadata from the Apple App Store during 39 weeks to identify the factors that affect an app's likelihood of staying in the top (most popular) charts, thus generating more revenue. They collected information for 17,697 apps, through time, getting a final count of about 530K observations. Palomba *et al.* [28, 29] used reviews from 100 apps to (i) trace informative reviews onto source code change, and (ii) monitor the extent to which developers accommodate crowd requests and follow-up user reactions as reflected in their ratings.

In addition to previous studies, Android-related datasets have been released. AndroidTimeMachine [9] is a graph-based database which contains metadata of 8431 open source Android apps hosted at GitHub. AndroidTimeMachine includes Git repositories with full commit history and metadata extracted from the Google Play store, such as app ratings and permissions to analyze the relationships between source code and metadata. However, AndroidTimeMachine does not have historical records of the apps, thus, it only has a snapshot of the apps. The Android Apps and User Feedback [11] is a large dataset of Android applications belonging to different apps categories, which includes users feedback, documentation of the evolution of the related code metrics and more than 6 hundred app versions. Finaly, AndroZoo[5] collects a set of more than 10M APKs extracted from different app' markets, for malware detection. AndroZoo stores metadata but related to APKs (*e.g.,* APK compilation date, SHA256 hash, etc.)

It is worth noting that the papers and datasets presented above provide useful information for analyses, however, even when some of them take into account changes over time, none of them is purely focused on the GP, its metadata and its changes over different periods of time. Even when, some of them use top apps' metadata, they are focused on factors that improve sales performance, while **`Hall-of-Apps`** includes top apps with the aim of improving release engineering strategies to help developers to release successful apps.

5 FUTURE WORK

Potential Dataset Usage: We expect **`Hall-of-Apps`** [7] be used to conduct research in the field of release engineering using classification and prediction models to track survivability of apps in GP. Moreover, **`Hall-of-Apps`** could be used for longitudinal studies using Natural Language Processing methods to analyze apps descriptions and reviews; those studies can be focused on changes over time and how they affect the success of an app. In addition, because **`Hall-of-Apps`** has data for different time snapshots, potential usages could be in the field of trend analysis, survivability analysis, release engineering strategies, longitudinal market analysis. Note that **`Hall-of-Apps`** has information from different countries that could be used to identify specific features that allow apps to be successful in the given country. On top of that, **`Hall-of-Apps`** contains information categorized by free and paid top charts, allowing more complex analyses (*e.g.,* paid apps in a given country)

Dataset improvement: In order to improve our dataset, both the GP scraper and the HTML parser should be adapted to extract information from the current GP website, and increase the size of the dataset. In addition to the dataset, public tools will be released to analyze and perform more complete analyses. Furthermore, it would be desirable to parse raw data right after its extraction to reduce the amount of time it takes to process more than one week at a time. In addition, **`Hall-of-Apps`** can be complemented using previous datasets such as, AndroidTimeMachine [9] or RapidRelease [16], which contain metadata from GitHub to include a development perspective, as well as datasets which provide information about changes in permissions [30] and datasets.

Finally, **`Hall-of-Apps`** could be improved by extracting information over several months from different stores, such as App Store or unofficial Google Markets (*e.g.,* Anzhi or AppGallery) to compare top apps' metadata and to determine different success strategies depending on the market.

6 CONCLUSIONS

We have presented the **`Hall-of-Apps`** [7], a dataset of thousands of records with metadata of successful apps released in the Google Play Store collected over several weeks and for different countries. We make this dataset publicly available with the purpose of supporting different analyzes of app's behavior over several periods of time. Moreover, we create a comprehensive online appendix that includes (i) the source code of the tools used to extract, parse and visualize the data, (ii) dataset description , (iii) links to download raw HTML data and a MongoDB dump, and (iv) data visualizations. The online appendix is available at **https://bit.ly/2uIkXs8**.

ACKNOWLEDGMENTS

This work is partially founded by a Google Latin America Research Award 2018-2020.

REFERENCES

[1] [n.d.]. 5 Quick Stats About Developers (And What They Mean). https://www.stackoverflowbusiness.com/blog/5-quick-stats-about-developers-and-what-they-mean. [Online; accessed 2-February-2020].

[2] [n.d.]. CheerioJS. https://cheerio.js.org/.

[3] [n.d.]. Google Play: number of available apps 2009-2019. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Online; accessed 2-February-2020].

[4] [n.d.]. Hall-of-Apps Online Appendix. https://thesoftwaredesignlab.github.io/hall-of-apps-tools/.

[5] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2901739.2903508

[6] A. AlSubaihin, F. Sarro, S. Black, L. Capra, and M. Harman. 2019. App Store Effects on Software Engineering Practices. *IEEE Transactions on Software Engineering* (2019), 1–1. https://doi.org/10.1109/TSE.2019.2891715

[7] Laura Bello-Jiménez, Camilo Escobar-Velásquez, Anamaria Mojica-Hanke, Santiago Cortés-Fernandéz, and Mario Linares-Vásquez. 2020. *Hall-of-Apps: The Top Android Apps Metadata Archive*. https://doi.org/10.5281/zenodo.3716367

[8] Rishi Chandy and Haijie Gu. 2012. Identifying Spam in the IOS App Store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality (WebQuality '12)*. Association for Computing Machinery, New York, NY, USA, 56–59. https://doi.org/10.1145/2184305.2184317

[9] Franz-Xaver Geiger, Ivano Malavolta, Luca Pascarella, Fabio Palomba, Dario Di Nucci, and Alberto Bacchelli. 2018. A Graph-Based Dataset of Commit History of Real-World Android Apps. In *Proceedings of the 15th International Conference on Mining Software Repositories (MSR '18)*. Association for Computing Machinery, New York, NY, USA, 30–33. https://doi.org/10.1145/3196398.3196460

[10] M. Goul, O. Marjanovic, S. Baxley, and K. Vizecky. 2012. Managing the Enterprise Business Intelligence App Store: Sentiment Analysis Supported Requirements Engineering. In *2012 45th Hawaii International Conference on System Sciences*. 4168–4177. https://doi.org/10.1109/HICSS.2012.421

[11] Giovanni Grano, Andrea Di Sorbo, Francesco Mercaldo, Corrado A. Visaggio, Gerardo Canfora, and Sebastiano Panichella. 2017. Android Apps and User Feedback: A Dataset for Software Evolution and Quality Improvement. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics (WAMA 2017)*. Association for Computing Machinery, New York, NY, USA, 8–11. https://doi.org/10.1145/3121264.3121266

[12] Emitza Guzman and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*. IEEE, 153–162.

[13] Theo Haerder and Andreas Reuter. 1983. Principles of Transaction-Oriented Database Recovery. *ACM Comput. Surv.* 15, 4 (Dec. 1983), 287–317. https://doi.org/10.1145/289.291

[14] 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 Press, 41–44.

[15] T. Johann, C. Stanik, A. M. A. B., and W. Maalej. 2017. SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 21–30. https://doi.org/10.1109/RE.2017.71

[16] Saket Dattatray Joshi and Sridhar Chimalakonda. 2019. RapidRelease: A Dataset of Projects and Issues on Github with Rapid Releases. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR '19)*. IEEE Press, 587–591. https://doi.org/10.1109/MSR.2019.00088

[17] Jieun Kim, Yongtae Park, Chulhyun Kim, and Hakyeon Lee. 2014. Mobile application service networks: Apple's App Store. *Service Business* 8 (03 2014). https://doi.org/10.1007/s11628-013-0184-z

[18] Zijad Kurtanović and Walid Maalej. 2017. Mining user rationale from software reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 61–70.

[19] David Lavid Ben Lulu and Tsvi Kuflik. 2013. 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)*. Association for Computing Machinery, New York, NY, USA, 297–306. https://doi.org/10.1145/2449396.2449434

[20] Gunwoong Lee and Raghu Santanam. 2014. Determinants of Mobile Apps' Success: Evidence from the App Store Market. *Journal of Management Information Systems* 31 (10 2014), 133–170. https://doi.org/10.2753/MIS0742-1222310206

[21] Fabio Manenti, Stefano Comino, and Franco Mariuzzo. 2015. Updates Management in Mobile Applications. iTunes vs Google Play. https://doi.org/10.13140/RG.2.1.1895.9528

[22] Daniel Martens and Walid Maalej. 2019. Towards understanding and detecting fake reviews in app stores. *Empirical Software Engineering* 24, 6 (01 Dec 2019), 3316–3355. https://doi.org/10.1007/s10664-019-09706-9

[23] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang. 2015. The App Sampling Problem for App Store Mining. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 123–133. https://doi.org/10.1109/MSR.2015.19

[24] William Martin, Federica Sarro, and Mark Harman. 2016. Causal Impact Analysis for App Releases in Google Play. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. Association for Computing Machinery, New York, NY, USA, 435–446. https://doi.org/10.1145/2950290.2950320

[25] W. Martin, F. Sarro, Y. Jia, Y.Zhang, and M. Harman. 2017. A Survey of App Store Analysis for Software Engineering. *IEEE Transactions on Software Engineering* 43, 9 (Sep. 2017), 817–847. https://doi.org/10.1109/TSE.2016.2630689

[26] Shahab Mokarizadeh, M.T. Rahman, and Mihhail Matskin. 2013. Mining and analysis of apps in google play. (01 2013), 527–535.

[27] Andreas Möller, Florian Michahelles, Stefan Diewald, Luis Roalter, and Matthias Kranz. 2012. Update Behavior in App Markets and Security Implications: A Case Study in Google Play. 3–6.

[28] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. 2015. User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 291–300. https://doi.org/10.1109/ICSM.2015.7332475

[29] Fabio Palomba, Mario Linares-Vásquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. 2018. Crowdsourcing user reviews to support the evolution of mobile apps. *Journal of Systems and Software* 137 (2018), 143 – 162. https://doi.org/10.1016/j.jss.2017.11.043

[30] Gian Luca Scoccia, Anthony Peruma, Virginia Pujols, Ben Christians, and Daniel E. Krutz. 2019. An Empirical History of Permission Requests and Mistakes in Open Source Android Apps. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR '19)*. IEEE Press, 597–601. https://doi.org/10.1109/MSR.2019.00090

[31] 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 (ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 14–24. https://doi.org/10.1145/2884781.2884818