A Recommender System for Mobile Applications of Google Play Store

Ahlam Fuad[1], Sahar Bayoumi[2], Hessah Al-Yahya[3]
Department of Information Technology
College of Computer and Information Sciences
King Saud University, Riyadh, Saudi Arabia[1, 2, 3]
Institute of Graduate Studies and Research,
Alexandria University, Alexandria, EGYPT[2]

Abstract—With the growth in the smartphone market, many applications can be downloaded by users. Users struggle with the availability of a massive number of mobile applications in the market while finding a suitable application to meet their needs. Indeed, there is a critical demand for personalized application recommendations. To address this problem, we propose a model that seamlessly combines content-based filtering with application profiles. We analyzed the applications available on the Google Play app store to extract the essential features for choosing an app and then used these features to build app profiles. Based on the number of installations, the number of reviews, app size, and category, we developed a content-based recommender system that can suggest some apps for users based on what they have searched for in the application's profile. We tested our model using a k-nearest neighbor algorithm and demonstrated that our system achieved good and reasonable results.

Keywords—Application profile; content-based filtering; Google play; mobile applications; recommender systems

I. Introduction

Recent years have witnessed massive growth in mobile devices with an increasing number of users as mobile devices have become part of every component of modern life. The smartphone market has grown dramatically, and users can now take advantage of various features in applications, which can easily be obtained from centralized markets, such as Google Play. Google Play is Google's official store and portal for Android apps that was launched in 2008 and accumulated more than 1 million downloadable and ratable applications now [1]. In December 2018, the number of available apps in the Google Play App Store was nearly 2.6 million [2].

Due to the substantial and growing number of available mobile applications in application stores, it becomes necessary to provide a system that identifies user interest based on what the system believes the user likes through his/her profile. Using a user profile would support an efficient and personalized application filtering system.

The general idea of filtering is to get a sub-collection of applications based on a specified category. There are different approaches to performing information filtering, including classification and recommendation. Classification is a step taken to reduce the sparseness of the input space by classifying applications into predefined interest categories. The applications in stores are labeled according to a high-level and store-specific classification method. This approach is limited by the fact that it depends entirely on the textual description available from the store [3].

Moreover, a recommender system is another way to filter information and is widely used in several domains. It is a decision-making tool that helps developers predict what a user will like or dislike from a list of applications. It provides personalized information by learning the user's interests from tracing through his/her interactions. It is also an excellent option for search fields as a recommender system that lets users discover more applications [3].

In this work, we explore a method of constructing recommender systems for apps in the Google Play app store based on the app profile. Issues related to modeling app preferences and choosing a set of recommended apps were investigated. Furthermore, a k-nearest neighbor classification approach (KNN) to classify apps based on the most influential attributes of apps within categories proposed. A prototype system is then built as a proof of concept, which tracks application profiles and then presents recommended applications to the user. Therefore, the research aim to answer the following question:

What are the most significant attributes of an application profile that could be used for developing a recommender system?

The remaining sections of this paper are organized as follows. Section II presents an overview and background of the topic. Section III discusses the related work of analyzing apps in app stores and app recommender systems. In Section IV, the methodology is introduced, and the results are explained and discussed in Section V. Finally, conclusions and future work are presented in Section VI.

II. Background

In this section, we discuss the background information and knowledge domains required for developing a recommender system.

A. Pearson's Correlation Coefficient

Pearson's correlation coefficient is also referred to as Pearson product-moment correlation coefficient (PPMCC) or the bivariate correlation, is a measure of the linear correlation between two variables [4]. It evaluates how well the

relationship between two variables can be described. The statistic defined in the range [–1, +1] which indicates how strongly the two variables are associated, where -1 indicates total negative linear correlation and 1 indicates total positive linear correlation. A value of 0 indicates no correlation.

B. K-Nearest Neighbor Classification (KNN)

The k-nearest neighbors' algorithm is a type of instance-based supervised learning approach. It is one of the simplest and most commonly used classification techniques and is easy to learn and implement and robust to noise. It is used mostly for classification and sometimes for predictive regression problems, in which a number of nearest neighbors of each data point are used based on the value of k, which represents how many nearest neighbors are to be considered to determine the class of a test sample data point. In other words, the KNN algorithm finds solutions by identifying similar objects. It is also called lazy learning because the function is only approximated locally and all computation is postponed until classification. This rule preserves the complete training set throughout the learning process and assigns to each query a class represented by the most frequent label of its KNN in the training set. One of the significant drawbacks of KNN is that becomes slow as the size of the data in use increases [5], [6].

C. Recommender Systems

Recommender systems (RSs) are techniques and software tools that provide users with suggestions for information or items that may be of interest to the user. Those suggestions will improve the user's decision-making processes, such as choosing what music to listen, what things to buy, or what apps to install. Thus RSs are the most popular and powerful tools in ecommerce [7]. Coincidence is one of the major stimuli for RSs to help the user discover things he did not look for explicitly. The essential computational task of RSs is predicting the subjective evaluation which the user gives to an item. These predictions can be computed by using predictive models with common characteristics. For example, the ratings of the user's previously purchased items can be exploited. Recommendation systems can be classified into three major categories to generate a list of recommendations based on a particular prediction technique [8], [9].

1) Content-based recommender systems: Content-based recommendation approaches analyze the descriptions of items rated by a user previously to build a user profile of his interests based on the items' features. Later, this profile will help to suggest additional items with similar properties. Content-based recommendation systems use methods that are focused on the items' characteristics or descriptions. These methods build a profile for each user, which is called a content-based profile that conserves the features of the previously viewed items. Then, the RS will get the most suitable details for the user by comparing the information in the generated profile and the descriptions of items [7]. For example, we have our items: A, B, C, and D. Tom likes items B, C, and D; John wants A, B, and C; and Sozy likes C. Therefore, by comparing John's and Tom's liked items, it is apparent that they both like B and C, and then the recommender system conclude that B and C are similar. If Sozy likes C, then item B should be recommended to him.

2) Collaborative recommender systems: Collaborative recommender approaches collect feedback information from all the users who rate the items. These approaches build a model based on the user's past behavior and the similar decisions of the other users. Thus, this model can be used to predict items the user may be interested. For example, Sozy again likes C and D. We need a recommender system to search for a person with similar preferences to Sozy, so we can notice that Tom also likes C and D. Therefore, he is the user who is identical to Sozy. Because he also likes B, B is recommended to Sozy.

Content-based approaches mostly perform better than collaborative filtering, especially when the data is extremely sparse. Merging both methods may improve the results {Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices}.

3) Hybrid recommender systems: Hybrid recommender systems have been developed by combining the abilities of both collaborative and content-based recommendations. These systems were introduced due to the limitations of the two techniques described above. Hybrid recommender approaches have been implemented using several methods: by applying the content-based and collaborative-based predictions separately and then combining both of them, by adding content-based capabilities to the collaborative-based approach (or vice versa), or by unifying the approaches into one model. Hybrid methods provide more accurate recommendations than simple approaches (collaborative methods and content-based methods).

III. Related Work

This section discusses the state of the art of within two directions: data analysis techniques and recommender systems. The related studies divided into three categories: analyzing applications in different app stores, applications based on similarity measures, and application based on recommender systems.

A. Application Analysis Studies

The author of the study [10] aimed to analyze app store data. They extracted feature information from a set of data collected from Blackberry apps using data mining in order to analyze the technical, business, and customer issues of apps. The results of this work indicate a strong correlation between the rank of app downloads and the customer rating and no relationship between price and rating, nor between price and downloads. These results partially match those observed in [11], where the study aimed to analyze the Google app store in order to identify correlations among app features, and the authors found a strong relationship between the number of downloads and price as well as between participation and price. In a recent study [12], the authors investigated the factors which impact the rating of Google play store apps. They analyzed 10,840 apps, and they indicated that app ratings help to get more downloads. Furthermore they found

that the used keyword in the app title plays an important role in determining the higher and lower ratings.

Studies [13] and [14] aimed to analyze the characteristics of apps extracted from app stores. Interestingly the experiments of [13] proved that the app size, the number of promotional images displayed on the app's web store page, and the app SDK version are the most influential factors in defining high-rated apps. Studies [1] and [15] used the Causal Impact Release Analysis tool to facilitate app store analysis.

Studies [16] and [17] introduced a novel approach for app classification utilizing features extracted from both web knowledge and relevant real-world context. Then, they integrated these extracted features into a machine learning model (Maximum Entropy (MaxEnt)) for training an app classifier.

B. App Similarity Studies

The study [18] introduced a classification system in order to classify mobile apps. They mined 5,993 apps from both the Apple and Google app stores and then classified them based on support vector machines (SVMs). As a result of this study, the automated app classification system achieved a excellent accuracy. Another study [19] proposed a novel technique for measuring the similarity among apps based on agglomerative hierarchical clustering techniques. They mined data for 17,877 apps from the Google and BlackBerry app stores. The empirical results of this study indicate an improvement over the existing categorization quality of both stores. In another study [11], the authors aimed to build clusters of similar apps using a probabilistic topic modeling technique and a k-means clustering method. The results showed that the Google Play categorization system does not respect application similarity.

The study [20] addressed the application classification issue and introduced a novel method for classifying apps using two methods. The first method used a neural language model applied to smartphone logs to embed apps into a low-dimensional space, while the second one used the k-nearest neighbors' classification method in the embedding space; the experimental results showed that the second proposed approach outperformed the current state of the art.

In a recent study [21], the authors introduced a classification method for local mobile app using deep neural network. They evaluated a dataset of Google Play to demonstrate the effectiveness of their method. Their results outperformed the baseline method by 5.5% related to F1 score. This study focused just on classifying local apps such as "Travel & Local" in the store.

A new framework for app categorization (FRAC+) has been proposed in [22], which is based on a data-driven topic model to suggest the appropriate categories for an app store, as well as to detect miscategorized apps. Experiments with the proposed system have shown that it is aligned with the new categories of the Google Play store.

C. Application Recommender System Studies

There is considerable literature available on both recommendation systems and mobile recommendation systems with various descriptions of recommendation systems in general [1], [23], [24]. The authors in [25] discussed the incorporation of recommender systems in the mobile application domain. They used a hybrid recommender system to deal with the added complexity of context and recommend appropriate mobile applications to users. Thus, this approach provides positive ratings. Therefore, based on this study, users can select from among several content-based or collaborative filtering components.

A new efficient framework called "SimApp" was proposed to detect similar applications using an online kernel learning algorithm [26]. They crawled real data from the Google app store and extracted a multi-modal heterogeneous data set. Their outcomes indicate the efficiency of the proposed framework. The similarity of items may help the application of content-based recommender systems. Another study introduced a framework based on the incorporation of version description features into app recommendation [27]. Another study [28] described the implementation of a hybrid recommender system that employed five different filtering techniques to help users when choosing a new application to download from a market. This system was also able to solve many common problems found in collaborative recommender systems that reduce the quality of the generated predictions. The study was based on using information collected from different users to support users with recommendations based on their history. The results showed good performance in terms of mean absolute error (MAE) and users' satisfaction.

The study [8] discussed assisting the users in choosing the appropriate application using recommendations. The author proposed a recommender system for mobile applications by integrating two methods: tracking user behavior to get his preferences to find new and similar apps to their used ones and utilizing the user's context in order to provide him with useful recommendations by using the Google Play Engine. While in the study [29], the authors proposed a recommender method for apps based on graph techniques. Interestingly, the proposed method can recommend apps without the need for specifying user preferences. Another paper proposed a recommender system for the mobile application market by understanding the mobile user's preferences and usage patterns for the types of applications they select and the online downloading process. The authors collected data from Google Play and then used statistical analysis and a pilot survey to find app features that influence user choices [30]. In [31], the authors proposed a novel structural user choice model (SUCM) to learn fine-grained user preferences by exploiting the hierarchical taxonomy of apps (tree hierarchy of apps). Also, they designed an efficient learning algorithm to estimate the model parameters. They used a diverse dataset of 52,483 users, 26,426 apps, and 3,286,156 review observations. The outcomes of this study show that SUCM consistently outperforms state-of-the-art Top-N recommendation methods by a significant margin. The study in [32] proposed a novel method using a unified model that combines content-based filtering with collaborative filtering, harnessing information from both ratings and reviews. This study applied topic modeling techniques to the review text and aligned the topics with rating dimensions to improve prediction accuracy. Another study [33] proposed a unified model VAMF for the

version-aware mobile app recommendation problem to address the data sparsity issue by incorporating review text from both the version level and the app level and modeling version based correlations of version-level temporal correlations and app-level aggregate correlation. They also proposed an efficient algorithm to solve the model and analyze its optimality and complexity. They used a Google Play dataset that contained the reviews for all of its versions and the descriptions of its latest version. The experiments conducted in this study on a large dataset showed that the proposed method outperforms comparable methods in prediction accuracy and that the proposed algorithm can be linearly scaled.

The study in [34] introduced a sequential approach for modeling the popularity of mobile apps by collecting data from 15,045 apps. They produced a popularity-based hidden Markov model (PHMM) for a variety of tasks, including app recommendation and review spam detection, and demonstrated its usefulness in ranking fraud detection. The experimental results validated both the effectiveness and efficiency of the proposed popularity modeling approach. Another study built on a hidden Markov model where the authors proposed a mechanism for modeling three main factors governing the app installation behavior of smartphone users: short-term context, co-installation pattern, and random choice. Then, a heterogeneous hidden Markov model (heterogeneous HMM) was used to incorporate these main factors. They used a combination of app installation data from the installation records of 9009 users with a portion of the Netflix data set from 54,314 users on 3561 movies. The experimental results indicated that the proposed system can outperform other methods consistently under different experimental settings [35].

The study [36] was generally focused on recommending independent items to users who were suggested by a hybrid cross-platform app recommendation (STAR) system. Another study [37] introduced recommender systems on mobile platforms based on user profiles generated from the installed apps. They improved on existing machine learning models to predict user profiles. The results of this study showed an increase in these models' predictive accuracy. Furthermore, study [38] introduced a recommender system (Vanilla) that considers social and contextual information processes. The system allows the comparison of different recommendation techniques. Besides this, Vanilla includes eleven contextual dimensions and a mechanism for analyzing the influence of social networks on app consumption. They found that the new proposed approach has a strong correlation with previous approaches and better efficiency than other techniques. A recent study proposed a context-aware approach for mobile app recommendation using tensor analysis (CAMAR) [39]. They conducted data analysis on Google Play Store and Apple's App Store in order to find the mobile apps characteristics. They utilized an effective tensor-based framework to integrate the features on users and apps and app category information to facilitate the app recommendation performance. Thus, they demonstrated the effectiveness of their proposed method.

A considerable amount of literature has been published recently regarding recommender systems for mobile apps. One study proposed a recommender system for mobile apps based on user reviews using topic modeling techniques and probability distributions to represent apps features [40]. Hence, this study aimed to construct a user profile based on his installed apps in order to identify his preferences. Therefore, they found that user reviews, extracted from datasets that were crawled from the Apple App Store, represented apps features efficiently. Another study [41] introduced a mobile sparse additive generative model (Mobi-SAGE) to recommend apps. They crawled an extensive collection of apps from the 360 App Store in China. The results of their study demonstrated that the proposed model outperformed other existing state-of-the-art methods.

According to the literature review, many models were developed using variety of features to support users selecting applications. Table I show a summary for researches discussed in the last section regarding the platform and used features for deployed recommender systems.

TABLE I. SUMMARY OF RECOMMENDER SYSTEM RESEARCHES

Ref.	Platform	Attributes
[25]	play.tools framework	Ratings.
[26]	Android	Name, category, description, developer, update, permissions, and app logo/images.
[27]	Apple	Version-categories, genre, and ratings.
[28]	Android, Apple	User history, Tags used to define the applications by the user, and user satisfaction.
[29]	Apple, Android, Blackberry and Windows App store	Apps installed on the user's phone.
[30]	Android	Cost, app logo/ image, gender, and types of downloaded applications.
[31]	Android	Category tree.
[32]	Amazon Dataset	Ratings and reviews.
[33]	Android	# of users, # of versions, and # of ratings.
[34]	Apple	Trend based Applications, rating, review spam detection, and ranking fraud detection.
[35]	Android	User installation behavior, user preferences, and Modeling random choice.
[36]	Apple	Application Rating between different platforms (iPhone-iPad platform and iPhone-iPad-iMac platform).
[37]	Android	Cost, ratings, and user profile based on the installed applications.
[38]	Android	Categories and ratings.
[39]	Android, Apple	User's preference, app category, and features of multiple views.
[40]	Apple	User preferences and reviews.
[41]	Android	User interests, ratings, and privacy preferences.

As shown, most of previous researches used either collaborative or hybrid approaches for building recommender systems which increase system complexity.

Our research aims to study the inter-relation between app attributes to select the most significant features. Furthermore, develop a content-based recommender system using the selected attributes.

IV. Methodology

In this section, we illustrate our proposed system, which includes five steps, as shown in Fig. 1. In the first step, we acquired the dataset. In the second step, we prepared the dataset for analysis to complete the other steps. In the third step, we analyzed the data to identify the correlations among the features. In the fourth step, we designed a suitable recommender system model. The final step is to test the recommender system model and make some recommendations.

The rest of this section explains the steps of our approach in more detail.

A. Data Acquisition

The dataset used to achieve this study is consists of 10841 apps scraped from the Google Play store, which publicly available on the Kaggle website [42], where its most recent update provided two months ago. This dataset provides detailed information from Google about the apps on the Google Play store. Thus, it includes 13 attributes with three datatypes: String, Categorical and Numeric. Only the "reviews" belongs to the numeric data type with values range from 0 to approximate 78Million reviews. Table II shows the attributes based on data types.

The dataset includes 33 different categories (shown in Fig. 2) and 118 genres, which define the sub-category for each application.

An initial statistical summary about the numerical data shown in Table III to provide a deep understanding about each attribute for further processing.

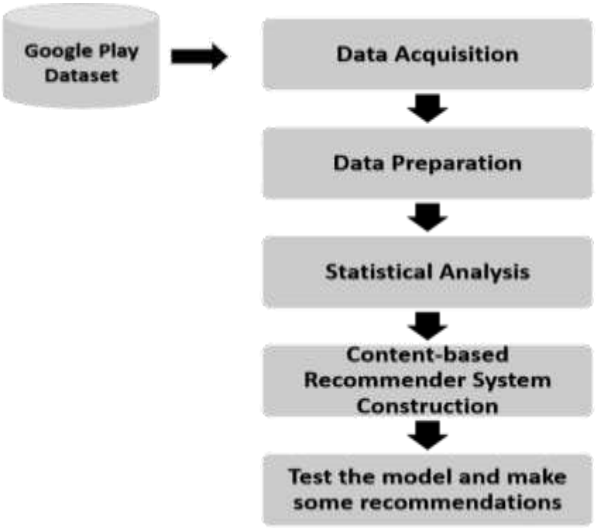

Fig. 1. Proposed System Methodology.

TABLE II. The Google Play Store Attributes Classified by Data Type

	Categorical attributes	
Installs	Type	Genres
Category	Content Rating	Android Ver.
	Rating	
	String attributes	
App name	Size	Price
Last Update	App Ver.	
	Numeric	
	Reviews	

Fig. 2. A Cloud Showing the Categories.

TABLE III. Statistical Summary for the Numerical Attributes

	Count	**Mean**	**STD**	**Min**	**Max**
Rating	8196	4.17E+00	5.37E-01	1.00E+00	5.00E+00
Reviews	9660	2.17E+05	1.83E+06	0.00E+00	7.82E+07
Size	9660	3.18E+07	3.48E+07	8.70E+03	1.05E+08
Installs	9660	7.78E+06	5.38E+07	0.00E+00	1.00E+09
Type	9660	7.82E-02	2.68E-01	0.00E+00	1.00E+00
Price	9660	1.10E+00	1.69E+01	0.00E+00	4.00E+02

B. Data Preparation

Data preparation is a necessary step to analyze the data correctly, and to facilitate understanding of the relationships among the data and to gain useful insights. As shown from Table III; the Kaggle dataset contains some missing values for "rating" attribute. In addition, inconsistencies in the attributes "size", "installs", "price" and "reviews" by remove unwanted information.

The data preparation process of the dataset using Python are applied in three consecutive steps as follows:

1) Removed duplicated rows which reduce the dataset by 1181 records.

2) Convert string and categorical datatypes into numeric to allow further data analysis.

3) Insure consistency of numeric attributes through the following:

a) Convert the characters 'K' and 'M' within the "Size" attribute to a numeric values.

b) Propagate last valid observation forward using forward fill method to replace the “Varies with device” value to get a numeric value within the “Size” attribute.

c) Convert the characters '+', '$', and 'M' from “Installs,” “Price,” and “Reviews,” into numeric values.

d) The last attribute that needed to be converted was “Type,” where we mapped the string values to numeric ones.

Regarding the missing values for the "rate" attributes, a null value kept for the associate records as they represent 15.15% of the total dataset. There are different reasons for the missing values within the "rate" attributes such as: new released application or not common for users. Therefore, we decided to include the records with the null value.

C. Statistical Analysis

Pearson’s correlation coefficient is used to measure the linear correlation between the numerical features of the apps in the Google Play store. Pearson’s correlation coefficient is a statistical measure used to determine the strength of the relationship between paired data [4]. Fig. 3 shows the Pearson’s correlation coefficient heatmap between the numeric features for the dataset. The correlation coefficients between attributes is the ground truth that help in choosing the most prominent features for further use in building the recommender system.

According to the heatmap; the attributes "reviews", "size", and "installs" are the most correlated attributes while other attributes are not.

D. Recommender System Construction

When looking for app a common attribute to be specified is the category. Then, a list of all apps under the specified category are shown. Sorting the apps to guide you to the best is restricted by choose one attribute. Based on the correlation coefficient and the importance of category attribute for the user; we decided to include all the four attributes ("reviews", "size", "installs", and "category") in defining the app profile for a content-based recommender system. The app profile consists of 37 columns; the first column is the app id within the dataset, and the next 33 columns represent the 33 categories of apps and three columns for "reviews", "installations", and "size". Furthermore, each column/feature scaled by its maximum absolute value for efficient calculations.

Fig. 3. Pearson's Correlation Coefficient Heat Map.

E. Classification using K-NN

A K-nearest neighbors (K-NN) algorithm; as an unsupervised machine learning; is used to measure the similarity between apps using their profiles. The nearest neighbor algorithm uses a “brute” algorithm and “cosine” metric.

V. Results and Discussion

Our proposed recommender system developed based on building a profile for each app using the most significant attributes. However, Pearson's correlation coefficient (as in Fig. 3) showed that "reviews", "size" and "installs" are the most significant correlated positively attributes. The highest correlated pair is “Installs” and “Reviews” with value 0.63. Thus, obviously highlight that users prefer to download apps that intensively reviewed. Fig. 4 shows a log scale for the relationship between the "installs" and the "reviews".

The second significant correlation between “Size” and “Installs” with value 0.19 shows a considered level of importance of the application size for users. The log scale relationship (Fig. 5) shows increase number of installs for small size applications while still large applications attract users.

However, most popular mobile apps, especially game apps, tend to be feature-rich, which implies that additional code and assets can pump up file sizes. Generally, the statistic indicates an increase in the number of mobile game app downloads from Google Play worldwide. In 2018, a total of 29.4 billion mobile games were downloaded globally across Google’s app store [43]. For example, the famous PUBG Game, which is sized at 1.6 GB for Android, is considered to be the most downloaded mobile game in the last quarter, which is a free, high-resolution game with excellent graphics and details [44].

Furthermore, a less significant correlation between “Size” and “Reviews” with a value 0.16 showed the importance of the "Size" attribute along with the "reviews" which highlight the user's need to optimize their storage use. Fig. 6 shows that apps with small sizes are with more reviews, which therefore more installs.

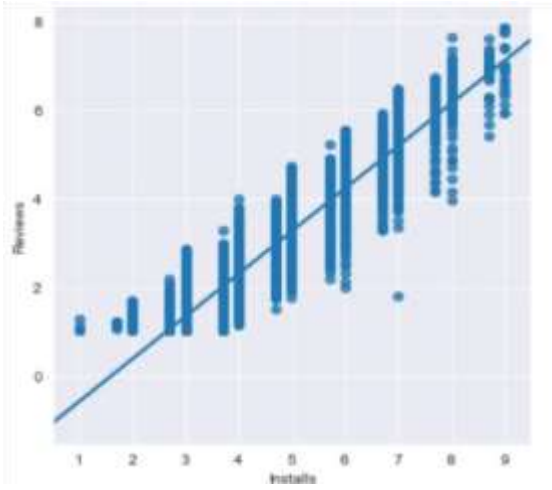

Fig. 4. Correlation between Installs and Reviews.

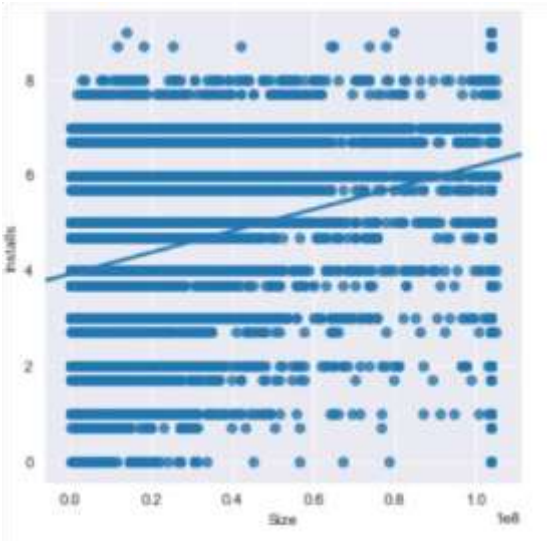

Fig. 5. Correlation between Size and Installs.

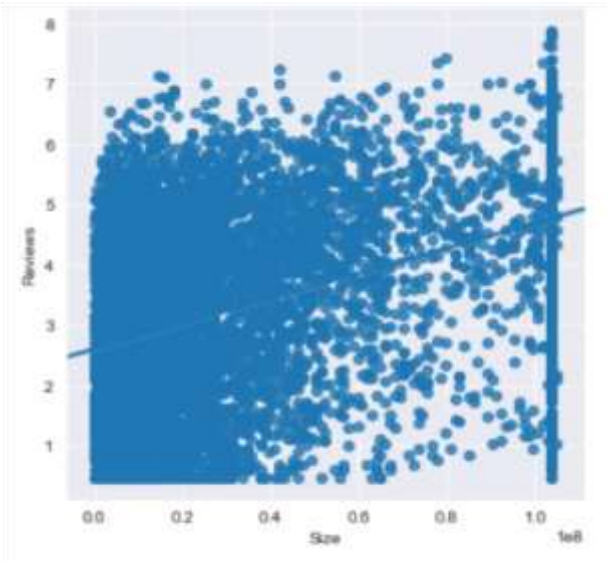

Fig. 6. Correlation between Size and Reviews.

To evaluate the developed recommender system; three case studies are used for different categories.

Case study 1: From the Social category; the “Facebook” application with id “8823”. By applying the “K-NN” to the matrix of profiles of all applications, four apps are recommended and sorted based on their K-NN metrics. Table IV shows the recommended apps and their distances from the Facebook app.

Case study 2: From the Game category, the “Candy Crush Saga” application with id “7484” is used for testing. By applying the “K-NN”, four recommended games are shown on Table V along with their K-NN distances.

Case study 3: From the education category, the "Wikipedia" application with id “8452” is used for testing. Table VI shows the recommended apps and their corresponding distance from the "Wikipedia" app.

TABLE IV. RECOMMENDED APPS FOR FACEBOOK CASE STUDY (ID=8823)

ID	App	Reviews	Size	Installs	distance
8824	Instagram	6.66E+07	1.04E+08	1.00E+09	4.86E-04
8827	SnapChat	1.70E+07	1.04E+08	5.00E+08	3.68E-03
3325	Facebook Lit	8.61E+06	1.04E+08	5.00E+08	4.81E-03
8830	Google+	4.83E+06	1.04E+08	1.00E+09	5.86E-02

TABLE V. RECOMMENDED APPS FOR CANDY CRUSH SAGA CASE STUDY (ID=7484)

ID	App	Reviews	Size	Installs	Distance
7485	Dream League Soccer2018	9.88E+06	7.76E+07	1.00E+08	1.21E-02
6947	Temple Run 2	8.12E+06	6.50E+07	5.00E+08	2.55E-02
8762	My Talking Tom	1.49E+07	1.04E+08	5.00E+08	2.76E-02
7508	Subway Surfers	2.77E+07	7.97E+07	1.00E+09	5.93E-02

TABLE VI. RECOMMENDED APPS FOR WIKIPEDIA CASE STUDY (ID=8452)

ID	App	Reviews	Size	Installs	distance
9587	English Hindi Dictionary	3.84E+05	1.04E+08	1.00E+07	1.65E-02
8455	Dictionary-Merriam-Webster	4.54E+05	1.04E+08	1.00E+07	1.26E-01
9496	Dictionary	2.64E+05	1.04E+08	1.00E+07	1.76E-01
8463	Moon+ Reader	2.34E+05	1.04E+08	1.00E+07	1.87E-01

VI. CONCLUSIONS

Mobile app recommendation based on only application installation records is a challenging task. In this paper, we proposed a model that seamlessly combines content-based filtering with application profiles. Thus, we used a real-world app dataset from Google Play to analyze app information and then utilized the most effective content to build a content-based recommender system. Based on our results, the most influential factors in choosing an app are the number of installs, number of reviews, app size, and category. Finally, we introduced some examples to prove that our system achieved good and reasonable results.

VII. FUTURE WORK

Although our proposed recommender system was originally designed for app recommendation from the Google Play store, we believe it can also be applied to other stores as well as other domains, such as book recommendation, music recommendation, movie recommendation, and food recommendation. Therefore, we believe that some possible future studies using the same experimental set up are possible.

Also, we aim to build a benchmark dataset from various application stores which could be used by researchers for building AI systems and recommender systems.

ACKNOWLEDGMENT

This research project was supported by a grant from the "Research Center of the Female Scientific and Medical Colleges", Deanship of Scientific Research, King Saud University. The authors thank the Deanship of Scientific Research and RSSU at King Saud University for their technical support.

REFERENCES

[1] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," IEEE Trans. Softw. Eng., vol. 43, no. 9, pp. 817–847, 2017, doi: 10.1109/TSE.2016.2630689.

[2] "Google Play Store: number of apps 2018 | Statista." [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Accessed: 30-Mar-2019].

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, Jun. 2005, doi: 10.1109/TKDE.2005.99.

[4] Y. Mu, X. Liu, and L. Wang, "A Pearson's correlation coefficient based decision tree and its parallel implementation," Inf. Sci. (Ny)., vol. 435, pp. 40–58, Apr. 2018, doi: 10.1016/j.ins.2017.12.059.

[5] M.-A. Amal and B.-A. Ahmed, "Survey of Nearest Neighbor Condensing Techniques," Int. J. Adv. Comput. Sci. Appl., vol. 2, no. 11, 2011, doi: 10.14569/ijacsa.2011.021110.

[6] S. Bafandeh, I. And, and M. Bolandraftar, "Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background."

[7] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Recommender Systems Handbook. 2011, doi: 10.1007/978-0-387-85820-3.

[8] V. Viljanac, "RECOMMENDER SYSTEM FOR MOBILE APPLICATIONS," Multimed. Tools Appl., vol. 77, no. 4, pp. 4133–4153, Feb. 2018, doi: 10.1007/s11042-017-4527-y.

[9] R. G. De Souza, R. Chiky, and Z. K. Aoul, "Open source recommendation systems for mobile application," CEUR Workshop Proc., vol. 676, pp. 55–58, 2010.

[10] A. Finkelstein, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "Mining App Stores: Extracting Technical, Business and Customer Rating Information for Analysis and Prediction," UCL Res. Notes, vol. 13, p. 21, 2013.

[11] S. Mokarizadeh and M. Matskin, "Mining and Analysis of Apps in Google Play," no. January 2013, pp. 527–535, 2013, doi: 10.5220/0004502005270535.

[12] A. Mahmood, "Identifying the influence of various factor of apps on google play apps ratings," J. Data, Inf. Manag., vol. 2, no. 1, pp. 15–23, 2020, doi: 10.1007/s42488-019-00015-w.

[13] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What are the characteristics of high-rated apps? A case study on free Android Applications BT - IEEE International Conference on Software Maintenance and Evolution," pp. 301–310, 2015.

[14] M. Ali, M. E. Joorabchi, and A. Mesbah, "Same App, Different App Stores: A Comparative Study," Proc. - 2017 IEEE/ACM 4th Int. Conf. Mob. Softw. Eng. Syst. MOBILESoft 2017, pp. 79–90, 2017, doi: 10.1109/MOBILESoft.2017.3.

[15] W. Martin, F. Sarro, and M. Harman, "Causal impact analysis for app releases in google play," pp. 435–446, 2016, doi: 10.1145/2950290.2950320.

[16] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian, "Mobile app classification with enriched contextual information," IEEE Trans. Mob. Comput., vol. 13, no. 7, pp. 1550–1563, 2014, doi: 10.1109/TMC.2013.113.

[17] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian, "Exploiting Enriched Contextual Information for Mobile App Classification," pp. 1617–1621, 9781450311564.

[18] G. Berardi, A. Esuli, T. Fagni, and F. Sebastiani, "Multi-store metadata-based supervised mobile app classification," Proc. 30th Annu. ACM Symp. Appl. Comput. - SAC '15, pp. 585–588, 2015, doi: 10.1145/2695664.2695997.

[19] A. A. Al-Subaihin et al., "Clustering Mobile Apps Based on Mined Textual Features," Proc. 10th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas. - ESEM '16, pp. 1–10, 2016, doi: 10.1145/2961111.2962600.

[20] V. Radosavljevic et al., "Smartphone App Categorization for Interest Targeting in Advertising Marketplace," 2017, pp. 93–94, doi: 10.1145/2872518.2889411.

[21] K. Ochiai, F. Putri, and Y. Fukazawa, "Local app classification using deep neural network based on mobile app market data," 2019 IEEE Int. Conf. Pervasive Comput. Commun. PerCom 2019, pp. 186–191, 2019, doi: 10.1109/PERCOM.2019.8767416.

[22] D. Surian, S. Seneviratne, A. Seneviratne, and S. Chawla, "App Miscategorization Detection: A Case Study on Google Play," IEEE Trans. Knowl. Data Eng., vol. 29, no. 8, pp. 1591–1604, 2017, doi: 10.1109/TKDE.2017.2686851.

[23] A. Arampatzis and G. Kalamatianos, "Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices," ACM Trans. Inf. Syst., vol. 36, no. 3, pp. 1–28, 2017, doi: 10.1145/3125620.

[24] H. Cao and M. Lin, "Mining smartphone data for app usage prediction and recommendations: A survey," Pervasive Mob. Comput., vol. 37, pp. 1–22, 2017, doi: 10.1016/j.pmcj.2017.01.007.

[25] W. Woerndl, C. Schueller, and R. Wojtech, "A Hybrid Recommender System for Context-aware Recommendations of Mobile Applications," in 2007 IEEE 23rd International Conference on Data Engineering Workshop, 2007, pp. 871–878, doi: 10.1109/ICDEW.2007.4401078.

[26] N. Chen, S. C. H. Hoi, S. Li, and X. Xiao, "SimApp: A Framework for Detecting Similar Mobile Applications by Online Kernel Learning," Proc. Eighth ACM Int. Conf. Web Search Data Min., pp. 305–314, 2015, doi: 10.1145/2684822.2685305.

[27] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua, "New and Improved: Modeling Versions to Improve App Recommendation," Proc. 37th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr., pp. 647–656, 2014, doi: 10.1145/2600428.2609560.

[28] E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López, "Which App? A recommender system of applications in markets: Implementation of the service for monitoring users' interaction," Expert Syst. Appl., vol. 39, no. 10, pp. 9367–9375, Aug. 2012, doi: 10.1016/j.eswa.2012.02.131.

[29] U. Bhandari, K. Sugiyama, A. Datta, and R. Jindal, "Serendipitous recommendation for mobile apps using item-item similarity graph," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8281 LNCS, pp. 440–451, 2013, doi: 10.1007/978-3-642-45068-6_38.

[30] A. Yasin, L. Liu, R. Fatima, and W. Jianmin, "Designing the Next Mobile App Recommender System for the Globe," 2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC), 2017, pp. 491–500, doi: 10.1109/ISPAN-FCST-ISCC.2017.44.

[31] B. Liu, Y. Wu, N. Z. Gong, J. Wu, H. Xiong, and M. Ester, "Structural Analysis of User Choices for Mobile App Recommendation," ACM Trans. Knowl. Discov. Data, vol. 11, no. 2, pp. 1–23, Nov. 2016, doi: 10.1145/2983533.

[32] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in Proceedings of the 8th ACM Conference on Recommender systems - RecSys '14, 2014, pp. 105–112, doi: 10.1145/2645710.2645728.

[33] Y. Yao, W. X. Zhao, Y. Wang, H. Tong, F. Xu, and J. Lu, "Version-Aware Rating Prediction for Mobile App Recommendation," ACM Trans. Inf. Syst., vol. 35, no. 4, pp. 1–33, Jun. 2017, doi: 10.1145/3015458.

[34] H. Zhu, C. Liu, Y. Ge, H. Xiong, and E. Chen, "Popularity Modeling for Mobile Apps :," vol. 45, no. 7, pp. 1303–1314, 2015.

[35] V. C. Cheng, L. Chen, W. K. Cheung, and C. kuen Fok, "A heterogeneous hidden Markov model for mobile app recommendation," Knowl. Inf. Syst., vol. 57, no. 1, pp. 207–228, 2018, doi: 10.1007/s10115-017-1124-3.

[36] T.-S. Chua et al., "Cross-Platform App Recommendation by Jointly Modeling Ratings and Texts," ACM Trans. Inf. Syst., vol. 35, no. 4, pp. 1–27, 2017, doi: 10.1145/3017429.

[37] R. M. Frey, R. Xu, C. Ammendola, O. Moling, G. Giglio, and A. Ilic, "Mobile recommendations based on interest prediction from consumer's installed apps–insights from a large-scale field study," Inf. Syst., vol. 71, pp. 152–163, 2017, doi: 10.1016/j.is.2017.08.006.

[38] D. F. Chamorro-Vela et al., "Recommendation of Mobile Applications based on social and contextual user information," Procedia Comput. Sci., vol. 110, pp. 236–241, 2017, doi: 10.1016/j.procs.2017.06.090.

[39] T. Liang et al., "CAMAR: a broad learning based context-aware recommender for mobile applications," Knowl. Inf. Syst., vol. 62, no. 8, pp. 3291–3319, 2020, doi: 10.1007/s10115-020-01440-9.

[40] K. P. Lin, Y. W. Chang, C. Y. Shen, and M. C. Lin, "Leveraging Online Word of Mouth for Personalized App Recommendation," IEEE Trans. Comput. Soc. Syst., vol. 5, no. 4, pp. 1061–1070, 2018, doi: 10.1109/TCSS.2018.2878866.

[41] H. Yin, W. Wang, L. Chen, X. Du, Q. V. Hung Nguyen, and Z. Huang, "Mobi-SAGE-RS: A sparse additive generative model-based mobile application recommender system," Knowledge-Based Syst., vol. 157, pp. 68–80, 2018, doi: 10.1016/j.knosys.2018.05.028.

[42] "Google Play Store Apps | Kaggle." [Online]. Available: https://www.kaggle.com/lava18/google-play-store-apps/. [Accessed: 16-Mar-2019].

[43] "Global app stores mobile games downloads 2018 | Statistic." [Online]. Available: https://www.statista.com/statistics/661553/ global-app-stores-mobile-game-downloads/. [Accessed: 30-Mar-2019].

[44] "PUBG most downloaded mobile game last quarter, but revenue flags |GamesIndustry.biz." [Online]. Available: https://www.gamesindustry.biz/articles/2018-05-04-pubg-most-downloaded-mobile-game-last-quarter-but-fails-to-make-an-impact-with-revenue. [Accessed: 08-Sep-2020].