CTRS- a Recommendation Scheme for Apps Based on Topic Discoverability and Personality Matching

Jeeva R (✉ jeeva3710@gmail.com)

THAMIRABHARANI ENGINEERING COLLEGE

Dr. N. Muthukumaran

FRANCIS XAVIER ENGINEERING COLLEGE

Research Article

Keywords: Filtering, Pooling, App vector

Posted Date: January 27th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1269681/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Abstract

The recommendation of apps to the users is still a challenging task and differs from time to time. The past activities of the user alone don't help much in recommendation systems. It includes preferences, trends, location-based info, new arrivals, etc. For that, app features have to be correlated with the user profile to identify the apps for the recommendation. The reviews have been analyzed in two different ways. In the first part, hidden topics in reviews have been extracted that may be the app feature that has not been mentioned explicitly and app features can get updated. In the other part, user personality interests are discovered in the reviews and activities explored by the user with the known attributes like watch history, wish list, liked apps, previously installed apps, and trends in the platform. With CTRS, app attributes are extracted and an app vector has been created after the subsequent steps of filtering and pooling. It will be get updated in regular intervals due to the dynamic behavior of attributes like new features, current trends, rating, ranking, etc. From the derived app vector, the recommendation system decides whether to recommend an app or not when it encounters during the active session of the user. The experiment was conducted on an app store dataset from Kaggle over 2,54,303 apps in different categories, attributes, and reviews with polarity values. After analysis, the success rate of CTRS has been achieved up to 75% over higher and lower ranked apps in charts.

1. Introduction

The increasing development of mobile and the supported applications having a vital role in user's life due to simplicity, bearable price, and extensive features that are needed in the daily routine of many circumstances. The millions of apps are available to the users in the app store and the problem is how to deliver the specific app which the user desires. Most of the app stores suggest apps to the users either by no. of downloads or the comments of the user on that app. From one perspective, it makes a way for rising fraudulent apps by faking downloads and comments and on the other hand, the apps that are not desirable to the user are entered into the list. It is mandatory to design an RS that has to recommend the apps to the users based on their interest and should vary from time to time. The better RS not only favors the user but also the platform where the apps are laid out. More number of installs by the user in the platform increases the revenue and profit if the chosen RS is effective.

The parameters involved in RS are quite a lot. At the known level, RS comprises three entities: a user categorizer, a rank regenerator, and a framework to satisfy user needs. The user categorizer is based on a deep retrieval model that can analyze the apps in store among millions and recommend the better ones. For each app, a rank regenerator, i.e. a user tracking and preference model, predicts the user's activities along multiple dimensions with the help of parameters like downloaded apps, app usage time, browsing time over app and category, etc. Then make them as input to a Multi-target streamlining model whose output gives the most reasonable recommendations to the user.

App development has become one of the platforms for generating revenue and earning profits. Some free or low-graded apps were introduced in the store to raise the revenue of developers. As downloads have

been increased, it doesn't matter whether it works or satisfies the user. This type of one-trick applications is easily reachable to user due to the lack of visibility of apps that offers better service to achieve the objectives of the intended user. If the user's interests matched with the app characteristics through memory-based and model-based recommendation systems, the visibility of the genuine apps has been increased. Due to the dynamic nature of the model, the updates and the new arrivals influence the model only if it holds the primary features to attain the objectives of user preference [1].

App popularity plays a key role in app recommendation as it is based on rating, ranking, and reviews. Most recommendation systems follow this popularity attribute to match the user preferences to recommend apps. As it varies and sub-attributes are heterogeneous, popularity-based HMM has been used with trend-based recommendation systems. It calculates the rank γ_{rank} and average rating of the user γ_{rate} . On averaging the values with fusion factor α , a popularity score can be derived that not only is used to recommend apps but also indicate the apps trend in the future [2].

Most of the recommendation systems concentrate on user preference and app context. Though the role of the above attribute is big and dynamic, the relationship between the app and reviews has been not considered fully. Instead of deciding sentiments on review, evolve the process to determine the review similarity. As this process is iterative, the converging point of the process determines the relationship between the apps. If it didn't converge, the recommendation set can be revised with the above-obtained results. That not only produce better recommendation apps but is also used to calculate the relationship between the apps [3].

The rest of this paper is organized as follows. In Section 2, the evolution of app store recommendation techniques has been discussed. Section 3 presents the framework of a personalized recommendation system. Section 4 depicts the process flow of CNN based Topic Recommendation System (CTRS). Experiment results and analysis are shown in Section 5. The conclusions and further enhanced are given in Section 6.

2. Related Work

Matrix factorization is one of the popular methods in generating recommendations. So far, the accuracy has been strongly influenced due to the implicit feedback data as it needs to convert into user preference value. For that, RI-SGD has been deployed that trains the model first by including all the user factors and item factors to represent the confidence level. Next, the cost was assigned to each app in the store based on the training model and monitors for the entry for the new apps also. Finally, the incremental update model scans for new entries and recalculates the latent feature vectors to update the matrix instead of retraining the entire model. [4].

The popularity and development of mobile apps made a huge impact on human life. The recommendation of the apps is needed as the number of apps increasing rapidly in the store. The explicit mobile app data has not given sufficient information to decide the category or genre or rating etc. So the textual data of reviews given by the user not only point out the pros and cons but also the features of the

apps that are not mentioned in the app metadata. That information is “hidden topics” leveled by topic modeling and then fed into the recommendation engine along with the generated user profile. The intention is to discover apps with hidden features that may be preferred by the user which cannot be retrieved by app metadata. [5].

The implicit feedback data of app like usage time and browsing patterns thrives the recommendation system to deliver high-quality apps. But the processing of implicit data has a lot of challenges as each user’s browsing sequence has been not uniform. Traditional algorithms ignore the non-uniform behavior data that increases the possibility of losing hidden data that may reveal user’s interests. The differentiation of browsing behavior for each user is essential that indicate different interests. IMAR, a variant of LMR uses the browsing intensity to regularize the browsing patterns of the user which ranks the recommended apps based on recent interesting sequences and probability of highest download. [6].

The interpretation of user’s past behavior over apps has been done by many methods. CMARA formulate the user behavior trajectory that includes app usage data, time, location, and previous contextual information about the user. Based on the trajectory, the user similarity model has been constructed to identify the user with the same interests. The app list has been generated from the app list model that correlates with the location of the similar users by the active mining model. Then the list has been cross-referenced with the predicted preference to generate the recommendation list that includes the apps used by similar users and not considered as candidate apps by end-users. [7].

The review analysis of the app has been done by the five classifier algorithms to determine the sentiment polarity. The collected reviews from Google Play and the App store have been prepared for analysis using NLP techniques. The reviews are vectorized using the TF-IDF weighting technique and then data has been trained with 10 fold cross-validation. The polarity of the reviews is predicted by the classifiers like naïve Bayesian, random forest, SVM, Logistic regression, and SGD and the comparative analysis has been made over 600 mental health apps. The results depict the recommendation set which has higher positive polarity on the reviews. [8].

3. Personalized Recommendation System

An objective of a personalized recommendation system (PRS) is to recommend items that are desirable to users based on preferences and past activity that matches with queries. To access all information about the users, a semantic analysis is needed to influence the created user profile.

3.1 Data preprocessing

First, the user query has been processed by splitting it into keywords as shown in Fig. 1. Each keyword has been assigned a pos tag among the 87 Penn tree tag set corpus. After stop word elimination, polysemy words in the set have to be identified. For that, each word in the set has been fed into the synchronized word set with WordNet to check for a threshold value greater than 1. If yields put it in a framed word set and repeat the process for other words where the non-comply words.

3.2 User History Context

The user activity has to be collected from the unique identifier assigned to each item and user-level statistics. This data is very essential that makes to discover the items easily and increases the visibility of the recommendation systems. This UHC deployment was carried out in two phases: 1. Engagement 2. User Recommendation Process. The engagement parameter, average visiting time wtd is determined by taking the ratio of the sum of the total visiting time spent on the item to the total visit count. The wtd is normalized by the video length to express in the percentile that is reflected in both item and user profiles.

3.3 Recommendation process

The CNN, a type of feed-forward neural network recommends the items based on the features of an item and attributes of the users in the user profile. Each user has given a rating for different items at different times. Autoencoder reconstructs the rating matrix when the similarity set of user or item or both has been updated. Then, a similar rating of items matched with user preference has been collected by factorizing the item features with useful features. The similarity measure between the recommended items and others is identified to generate significant item recommendations.

4. Cnn Based Topic Recommendation System

4.1 Data collection

A dataset of over 2,54,303 apps of play store data has been collected that includes app name, id, rating, rating count, category, size, Status of Ad-Supported and In-App purchases, created date, last updated date, and minimum & maximum installs. From that, 64,295 apps have been filtered out based on the completion status of the above attributes. Then the reviews of apps have been checked for availability and excluded the apps that are not. 16,425 apps have been availed from the above filtering process with primary attributes and reviews with polarity values. The collected set is classified into two sets: 1. apps with 4 and above ratings are segregated with reviews AD_1 . 2. Apps with 3 and below ratings are segregated with reviews AD_2 .

4.2 Data preprocessing

As the app attributes are already organized except reviews, Natural Language Processing (NLP) techniques are applied to AD_1 and AD_2 . Symbols representing punctuations, special characters, and blank spaces will be removed. Duplicate words in a review that conveys the same meaning will be eliminated. Contractions in the reviews have to be identified and removed. Sentence cases in the reviews are converted into the common case. Word restoration is needed that reconstructs the words based on the similarity of words in the dictionary. After that, Stop words have to be identified and will be eliminated. Words in the review have to be converted into the root form using the Lemmatizer that narrows down to 14,605 reviews.

4.3 Feature Discovery

The KeySplitter extracts the keywords based on key base from reviews. As the dataset has been classified into two groups (AD_1, AD_2), the identified keywords are represented as N_1 and N_2 . Then for each app in the corpus (AD), the identified keywords were checked with the primary features of the category P_{feat} . If the keywords matched with the existing feature, they will be merged with P_{feat} . If it doesn't match and if it is a new feature for that app, add the feature to the P_{feat} of the app. If no feature has been identified, remove the specific review from the mined set.

Algorithm

Feature Discovery

Include AD_1 and AD_2

Input Reviews(R) and App Data, P_{Feat}

$N_1 = \text{KeySplitter}(\text{App_Reviews}_1)$

$N_2 = \text{KeySplitter}(\text{App_Reviews}_2)$

for each app in AD

$D = \{AD_1, AD_2\}$ belongs to AD

for each category in AD

$t = N_1, N_2$ belongs to D

if $t, P_{feat}(D, \text{App_domain}) > 0$ then

if t belongs to category

merge t with $P_{feat}(D)$

else

add t to $P_{feat}(D)$

end if

end if

end for

if $\text{update}(\text{App Data}, P_{feat}) < 0$ then

remove $N_1(u)$, $N_2(u)$ from D

end if

end for

4.4 Interest Mining from Reviews

Let $U = \{u_1, u_2, u_3 \dots u_n\}$ be the set of users, $AD = \{d_1, d_2, d_3 \dots d_n\}$ be the set of apps in the repository, $PB = \{p_1, p_2, p_3, p_4, p_5\}$ be the set of personalized behavior of users and $AR = \{R_1, R_2, R_3, \dots R_n\}$ be the set of reviews of apps in AD. The review R has been processed for mining hidden interests T by LDR. As we didn't expect the user gives to large reviews and LDR is not effective for one-liner reviews, apps visited or explored L_v by the user has been crawled. The semantic annotation extracts the topics that match with the behavior are collected and added to the user interest $I_T(U)$.

Algorithm

Interest Mining

Input: R, U, PB

Output: $I_T(U)$ (User's Topic Interest)

T, Topics of interest

$T \leftarrow \emptyset$

for each u in U

for each r in R

$T = \text{LDR}(r)$

if $T \neq \text{null}$ then

$L_v = \text{Visitedapps}(u)$

$T = \text{topicextraction}(\text{semantic_annotation}(L_v))$

end if

add T to $I_T(U)$

end for

end for

4.5 Recommendation Generation

In the CNN-based Topic Recommendation System (CTRS), the users in the platform have been fed into the activation function to generate the app vector for apps after the determination of $P_{\text{feat}}(\text{AD})$ and $I_T(\text{U})$ shown in Fig. 2. As shown in Fig. 3, App vector constitutes six key values that decide the recommendation status of the app. $P_F(\text{U})$ be the status of matching the app feature to the user's interest that takes the value 0 or 1.

PIM and PFM points the probability of interest and feature matched for the user and app. $C_{\text{AD}1}$ and $C_{\text{AD}2}$ represents the class of the apps belongs to the set AD_1 and AD_2 . Then the recommendation score for the app derived from the user interest is written as

User interest $\rightarrow$ App feature matches $P_F(\text{U}) = 0$ or 1

Probability of interest matched $\text{PIM} = 0$ to 1

Probability of feature matched $\text{PFM} = 0$ to 1

$C_{\text{AD}1}(\text{App in } \text{AD}_1) = 0$ or 1 ,

$C_{\text{AD}2}(\text{App in } \text{AD}_2) = 0$ or 1 , $\alpha = 0.5 \Rightarrow$ attenuation factor

Score ($P_{\text{feat}}, U|_T$) = $\sum_{k=1}^n (PIM_k(\text{AD}_1) * \alpha * PFM_k(\text{AD}_1))$

5. Experimental Analysis

5.1 Dataset Description

There are 2,54,303 apps in the dataset collected from the play store through GitHub that are updated till July 2021. It holds the information such as app id, category, rating, rating count, min and max installs, size, paid or free, price, minimum android version, developer id and website, released and last updated date, content rating, privacy policy, Ad-Support, In-App purchases, and editor's choice. App description and no. of reviews have been extracted through play_scrapper tool using app id. From this, 64,295 apps have been filtered out based on the completeness of the above attributes of the app. From the same source, 64,296 app reviews have been collected and cross over the filtered app's results in 16,425 apps. Then the apps whose reviews are not available or review is too short to process are eliminated through lemmatization results in 14,605 apps.

Next, Apps in the dataset are classified into 2 groups: Rating 4 and above (7526 apps) and Rating of 2 and below (1855 apps). In the two sets AD_1 and AD_2 , two categories of apps, "Communication" and "Education" have been retrieved that can be taken for training and testing samples for recommendation procedures.

5.2 Evaluation Metrics

The evaluation of the recommendation system is determined by the precision and recall of the retrieved apps. To determine the efficiency of PRS and CNN-based Topic-based RS, get the hidden topics from the description and reviews to be matched with a feature, matching the personality behavior of the user with review and success rate of recommendation apps. In this section, the number of apps that has been boosted with hidden features matched with the user's personalized behavior is termed as T_y , the number of apps that has been boosted with hidden features matched not matched with the user's personalized behavior is termed as T_n and the number of apps that matched without any boosting is termed as T_r . For that we have used the following evaluation metrics:

$$\text{Precision } (P_{ua}) := (T_y)/(T_y+T_n)$$

$$\text{Recall } (R_{ua}) := (T_y)/(T_y+T_r)$$

$$\text{F-Measure} := (2 * P_{ua} * R_{ua}) / (P_{ua} + R_{ua})$$

5.3 Results

The precision, recall and F-Measure of the PRS, and CTRS has been compared for the top apps in carts that belongs to the category of "Communication" and "Education" from the sets of AD_1 and AD_2 . We have taken 50 apps, 25 apps from AD_1 , and 25 apps from AD_2 .

The highest precision value for the app in AD_1 is recorded as 0.7812 for CTRS and 0.6487 for PRS as shown in Fig. 4(a). In Fig. 4(b), the highest recall value for the app in AD_2 is recorded as 0.7609 for CTRS and 0.6685 for PRS. From this, Fig. 4(c) reveals the F-Measure for PRS and CTRS has been computed and achieved a higher success rate of 75.85% for CTRS compared to PRS that 67.45%.

6. Conclusion

In this work, CTRS has been used to discover the hidden features for apps from the reviews and descriptions that increase the visibility of the app. Then the identification of personality behavior from reviews increases the relevant recommendations to the user. The success rate has been increased to 75% in recommending apps in terms of user behavior and the app's boosting matching features. In future work, grouping app and user similarity may increase the success rate of recommendation apps without increasing the false positives.

Declarations

This work was supported in part by Anna University Recognized Research Centre Lab at Francis Xavier Engineering College, Tirunelveli, Tamilnadu, India. Also, we would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. B. Erman, A. Inan, R. Nagarajan and H. Uzunalioglu, "Mobile applications discovery: A subscriber-centric approach," in *Bell Labs Technical Journal*, vol. 15, no. 4, pp. 135-148, March 2011.
2. H. Zhu, C. Liu, Y. Ge, H. Xiong and E. Chen, "Popularity Modeling for Mobile Apps: A Sequential Approach," in *IEEE Transactions on Cybernetics*, vol. 45, no. 7, pp. 1303-1314, July 2015.
3. M. Liu, C. Wu, X. Zhao, C. Lin and X. Wang, "APP Relationship Calculation: An Iterative Process," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2049-2063, 1 Aug. 2015.
4. C. Lin, L. Wang and K. Tsai, "Hybrid Real-Time Matrix Factorization for Implicit Feedback Recommendation Systems," in *IEEE Access*, vol. 6, pp. 21369-21380, 2018.
5. K. Lin, Y. Chang, C. Shen and M. Lin, "Leveraging Online Word of Mouth for Personalized App Recommendation," in *IEEE Transactions on Computational Social Systems*, vol. 5, no. 4, pp. 1061-1070, Dec. 2018.
6. He, Jiangning & Fang, Xiao & Liu, Hongyan & Li, Xindan. (2019). Mobile App Recommendation: An Involvement-Enhanced Approach. *MIS Quarterly*. 43. 827-849.
7. K. Zhu, Y. Xiao, W. Zheng, X. Jiao and C. -H. Hsu, "A Novel Context-Aware Mobile Application Recommendation Approach Based on Users Behavior Trajectories," in *IEEE Access*, vol. 9, pp. 1362-1375, 2021.
8. O. Oyeboade, F. Alqahtani and R. Orji, "Using Machine Learning and Thematic Analysis Methods to Evaluate Mental Health Apps Based on User Reviews," in *IEEE Access*, vol. 8, pp. 111141-111158, 2020.
9. Sami Hyrynsalmi, Tuomas Makil, Antero Jarvi, Arho Suominen, Marko Seppanen and Timo Knuutila.: App Store, Marketplace, Play! An Analysis of Multi-Homing in Mobile Software Ecosystems. *Proceedings of IWSECO 2012*, 59-72.
10. Gunwoong Lee and T. S. Raghu, Determinants of Mobile Apps Success: Evidence from App Store Market, *Journal of Management Information Systems*, Volume 31, 2014 - Issue 2, 1-46.
11. Caro-Alvaro, Sergio & García, Eva & García-Cabot, Antonio & de-Marcos, Luis & Gutiérrez-Martinez, Jose-Maria. (2017). A Systematic Evaluation of Mobile Applications for Instant Messaging on iOS Devices. *Mobile Information Systems*. 2017. 1-17. 10.1155/2017/1294193.
12. Wang, Haoyu & Liu, Zhe & Liang, Jingyue & Vallina-Rodriguez, Narseo & Guo, Yao & Li, Li & Tapiador, Juan & Cao, Jingcun & Xu, Guoai. (2018). Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets.
13. Jeeva. R, Muthukumaran. N, 'Detection of False Ranking Apps Using Level Aggregation', *Turkish Journal of Computer and Mathematics Education*, Vol. 12, No. 13 (2021), 2235-2242.
14. Jeeva. R, Muthukumaran. N, 'Identification of Fictitious Messages in Social Network using E-Hits and Newsapi', *International Journal of Innovative Technology and Exploring Engineering*, Vol. 9, Issue. 10, pp. 7- 11, August 2020.

15. Karthika. A, Muthukumaran. N, Joshua Samuel Raj. R, 'An Ads-Csab Approach for Economic Denial of Sustainability Attacks in Cloud Storage', International Journal of Scientific & Technology Research, Vol. 9, Issue. 04, pp. 2575-2578, April 2020.
16. Zahra, Fatima & Hussain, Azham & Haslina, Haslina. (2017). Usability evaluation of mobile applications; where do we stand?. AIP Conference Proceedings. 1891. 020056. 10.1063/1.5005389.
17. J. Rebekah, D. C. J. W. Wise, D. Bhavani, P. Agatha Regina and N. Muthukumaran, "Dress code Surveillance Using Deep learning," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 394-397, doi: 10.1109/ICESC48915.2020.9155668.
18. Chen, Ying & Xu, Heng & Zhou, Yilu & Zhu, Sencun. (2013). Is this app safe for children?: a comparison study of maturity ratings on Android and iOS applications. 201-212. 10.1145/2488388.2488407.
19. Zhu, Hengshu & Xiong, Hui & Ge, Yong & Chen, Enhong. (2015). Discovery of Ranking Fraud for Mobile Apps. IEEE Transactions on Knowledge and Data Engineering. 27. 74-87. 10.1109/TKDE.2014.2320733.
20. Hu, Bing & Liu, Bin & Gong, Neil & Kong, Deguang & Jin, Hongxia. (2015). Protecting Your Children from Inappropriate Content in Mobile Apps. 1111-1120. 10.1145/2806416.2806579.
21. Ali, Mohamed & Erfani, Mona & Mesbah, Ali. (2017). Same App, Different App Stores: A Comparative Study. 79-90. 10.1109/MOBILESoft.2017.3.
22. Martin, William & Sarro, Federica & Jia, Yue & Zhang, Yuanyuan & Harman, Mark. (2016). A Survey of App Store Analysis for Software Engineering. IEEE Transactions on Software Engineering. PP. 1-1. 10.1109/TSE.2016.2630689.

Figures

Figure 1

Architecture of PRS

Figure 2

Process flow of Feature Discovery

Figure 3

Framework of CTRS

4a

4b

4c

Figure 4

4(a). Precision of PRS and CTRS

4(b). Recall of PRS and CTRS

4(c). F-Measure of PRS and CTRS