Implementation of Agent-Based Games Recommendation System on Mobile Platforms

Pavle Skočir, Iva Bojić, and Gordan Ježić

University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, HR-10000 Zagreb, Croatia
{pavle.skocir,iva.bojic,gordan.jezic}@fer.hr

Abstract. Because of Google Play and App Store, today numerous different games are offered to every smartphone user. This diversity in supply is undoubtedly a good thing, but it also virtually disables users to find games they would like to play. However, it was shown that users tend to spend more money on purchasing recommended games when these recommendations are done using personal recommenders. In this paper we present an agent-based recommender for mobile platforms in which recommendations are made taking into account user game experience. Inputs for our recommender, which are collected both inside and outside the games, are stored in a semantic database. Based on collected information, user and game profiles are made that are then used in our recommendation algorithm. The focus of the paper is on how to do the implementation of the proposed system in real-world environments and obtain all the necessary data and how to make recommendations based on generated user and game profiles.

Keywords: Google Play, App Store, mobile games, user experience, user profiles, semantic database, MARS recommender system.

1 Introduction

Nowadays users are offered large amounts of content in form of applications for smartphones, videos on services like YouTube, or movies on IPTV platforms. In order to get users more familiar with the available content and to help them find what they might be interested in, service providers are developing personalized services that recommend specific content for each user. The available content recommendations can be made by using different methods. Generally, each recommendation system takes user interests as an input and generates personalized lists of recommended items for each user as an output [1].

Games as a content type require a higher level of user involvement compared to other content types. While playing games, users interact more with their devices than when, for example, listening to music. In the latter the only interactions that users have with their media players are for stopping, pausing, rewinding the content, while in the former various features about user interaction with games can be monitored. Such features include time needed for passing a certain level, collected points or touch gestures on smartphone screens.

In our previous work [2][3], we proposed an agent-based model for monitoring user experience through several parameters: user motivation, her/his progress through the game and how successful she/he was in order to find an appropriate genre and difficulty of the game which should be recommended to the user. Together with the appropriate genre and difficulty, our recommender also takes into account the download history and similarities between different games to provide users with new recommendations. This model has been implemented for a custom game [3]. In this paper, we focus on implementing our model in real-world environments, i.e. monitoring user interactions while playing existing games on smartphones and obtaining information about those games that were then used as inputs for our recommendation system. To the best of our knowledge, the approach of collecting information about users by monitoring user interactions while playing games is not so abundantly used like obtaining this information through ratings or reviews of purchased products [4].

The rest of the paper is organized as follows. Section 2 introduces the current achievements regarding recommendation systems, with focus on game recommenders. In Section 3 we present the model of our game provisioning system, while Section 4 describes the process of collecting inputs for our model from existing mobile platforms and within the existing games. Section 5 presents adaptation of our recommendation algorithm to enable working with the new set of data, different than originally proposed one. Section 6 concludes the paper.

2 Related Work

Even before App Store[1] and Google Play[2], it was clear that one-size-fits-all approach would not hold in case of mobile games and that recommendation systems would be needed in order to adapt gameplay [5]. General purpose recommendation systems can be classified in three main categories: content-based, collaborative and hybrid approach [6]. In content-based recommenders, users are recommended items similar to the ones they preferred in the past, while when using collaborative recommendations, users will be recommended items that other people with similar tastes and preferences to theirs liked in the past. Finally, a hybrid approach combines two aforementioned approaches.

Usually, there are three phases in every recommendation process: collecting the inputs, filtering the data and giving the outputs. Recommenders used for recommending games can gather inputs outside the games or inside the games [5]. When information about the games is collected outside, these services used for collecting are usually not strictly game-related and are used for plenty other purposes. On the other hand, when collecting information about users only inside the games, each game needs to be modified to collect the data. In our work we combine both approaches in order to achieve better results. Namely, we collect information about every specific game from App Store or Google Play, and combine them with information about user interaction inside the games.

[1] Initial release of App Store was on July 9, 2008.

[2] Initial release of Google Play (i.e. Android Market) was on October 23, 2008.

2.1 Collecting Inputs Outside the Games

Outside the games, information about users can be collected explicitly or implicitly. The explicit form of information collection relays on users giving their feedback, while the implicit way of collecting the data is mostly done without user knowledge. Nowadays, the most popular way to explicitly collect user opinions about the games they downloaded and tried is to collect their reviews and game ratings through App Store or Google Play. Both of these stores provide ranking for the top games in different game genres including both free and paid games, and for the top grossing games. Additionally, App Store provides information about best new games, current and previous editors' choices.

The easiest way to collect information about the users implicitly is to create their profiles that are then used to recommend personalized services. Roh and Jin developed the *Wallet App* for smartphones to gather personal information about the users [7]. In their system, each user profile consisted of static information such as user age, gender and address, as well as dynamic ones such as frequently used stores and favorite purchased items. Their goal was to develop a personalized advertisement recommendation system which would encourage users to spend more money on different apps. Namely, in 2009 Jannach and Hegelich [8] already showed that when using personalized recommendations instead of non-personalized ones a significant increase in viewed and sold items could be achieved. According to their results, effectiveness of online sale could be increased by up to 3.6 % by providing personalized item recommendations.

Which App? [9] is a general purpose recommender system that implements a hybrid approach (i.e. combines both techniques based on collaborative filtering and content-based filtering) and monitors user activities. The data that is being collected is: which applications are used, how many times the application has been used, during how much time it has been used, if it was used combined with other applications, how many times it has been updated, etc. Similar to work done by Roh and Jin, information collected through this app is also being used for building user profiles. However, unlike the *Wallet App*, this application collects information that can model user routines of using her/his smartphone better. The disadvantage is of course that this application has to run continuously in the background which increases battery consumption.

Another interesting way of collecting user opinion about the games they played implicitly was proposed in [10]. Sorensen based her approach on information collected from Twitter. Twitter is a service for a micro blogging where users can "follow" other users they are interested in. Before her research, much research that aimed at describing recommendations based on the relationships between users who were already "connected" on Twitter was done. However, her work did not leverage information about prior relations between different users, but was based on an overlap between words used by different users within the same game domain. In this way by extracting user tweets about different games, she managed to give valuable recommendations about unknown games to Twitter users within the same domain of interests without a prior knowledge about their relationships.

2.2 Collecting Inputs Inside the Games

Research conducted by Shin et al. [11] showed that fewer than a half of installed applications on mobile phones are actually being used, while Bohmer et al. showed that the average session with an application lasts less than a minute, even though users spend almost an hour a day using their smartphones [12]. It is thus very important to track user interactions in games in order to conclude whether users like them or not. Inputs inside the games can be also collected explicitly or implicitly. The easiest way to explicitly collect information about the user inside the games is to allow her/him to choose a difficulty level at the beginning of the game. This choice of a difficulty level is usually set on a single linear scale from easy to hard and it reflects user opinion of her/his abilities to successfully finish the game.

It is far more complex to implicitly collect information inside the games than outside them and to the best of our knowledge it was only done by Yang et al. in [13]. Yang et al. developed a mobile game recommender that was based on discovering of mobile games with similar gameplays taking into account player touch gestures while playing games. Gameplay is the specific way in which players interact with a specific game and the evaluation results from their work showed that touch gestures could serve as robust signatures of gameplay and that their recommender could give accurate recommendations of similar games simply based on users touch gestures. Consequently, common rank lists of the games or social recommendation approaches for discovering new games could be substituted or at least expanded with their system that allows players to search for games with a particular gameplay (e.g. slow-paced games for elderly people).

3 Agent-Based Model of Recommendation System

The model of our recommendation system called MARS - A Multi-Agent Recommendation System for Games on Mobile Phones is shown in Figure 3. In our model we used software agents because of their ability to automate business interactions and negotiate on behalf of their users. The model presented in this paper is a modification of our previous MARS model described in [3] and is based on three main processes: playing games, analyzing game consumption data and recommending games. While users are playing a certain game, User Agents monitor their interaction with the game. User interaction with the game determines the following parameters: user progress, motivation and success. These parameters model user experience and detailed description about how we chose and computed them can be found in [2,3]. After User Agents collect data about user interaction, they send the data to MARS Agent for analysis. Additionally, MARS Agent collects information about games (e.g. game genre, user ratings) from two most popular online stores: Google Play and App Store. The results of the analysis are used to recommend new games to User Agents who can then show these recommendations to users. If users like the recommended game, they can download it from Google Play or App Store and play it on their smartphones.

Data about games and user experience, inputs for MARS recommendation algorithm, is stored into a semantic database to enable logging of relationships among data which can be used later on for advanced reasoning. Based on these inputs, the recommendation algorithm calculates the list of games to be recommended specifically for each user. In order to collect the inputs, User Agents capture and send information about different events during playing of the game to MARS Agent: time moments when the whole game was started/finished and when different levels were started/finished. Additionally, MARS Agent collects information about games from different content providers (Google Play and App Store) and then calculates game recommendation lists for each user based on genres and difficulties of available games, the user motivation to play the game, user success in the game and progress that she/he is making in the game.

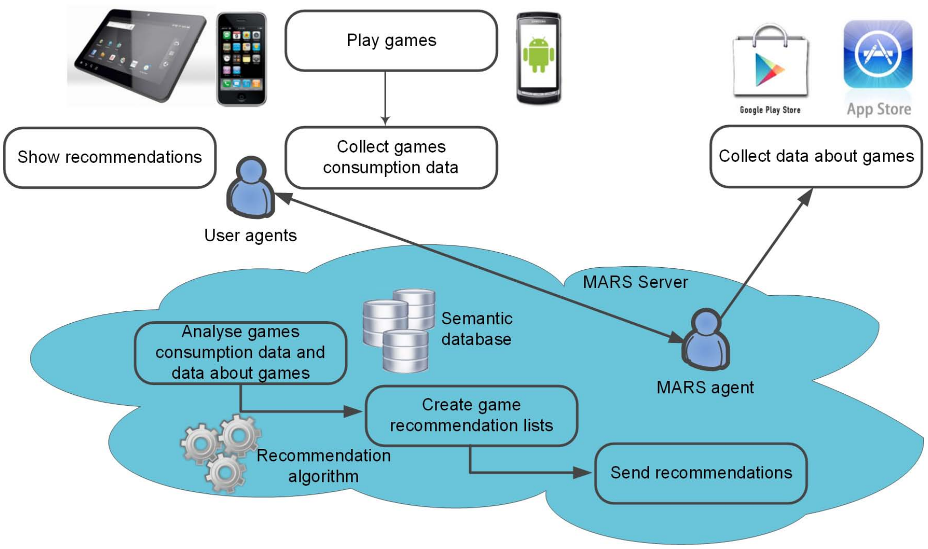

Fig. 1. The model of MARS recommendation system

4 Collecting Inputs for MARS Recommender

In order to collect data describing user experience, we had to monitor gameplay and based on user interactions within the game conclude how much she/he is motivated to play the game, how successful she/he is and whether she/he is making any progress. Moreover, we had to collect information about games from content providers. In this section we will describe how we collect information needed for our recommendation algorithm on two most popular mobile platforms - Android and iOS and its content stores Google Play and App Store. We want to recommend games regardless of the used platform, i.e. we want to use collected information from both platforms in order to produce game recommendations. To the best of our knowledge, this is a novel approach because currently game recommendations are performed only within the same platform.

4.1 Collecting Inputs Outside the Games

Outside the games, we collect information from Google Play and App Store. The recommendation algorithm that was previously proposed in [2] was based on information about genres and difficulties of the games. Both Google Play and App Store provide information about game genres, but it is not possible to collect information about game difficulties. Therefore, we needed to adapt our recommendation algorithm to give recommendations using the available information about games: e.g. game author, device platform, game genre, game price and rating. In this section we listed all available information about games from Google Play and App Store, and in Section 5 we discuss changes in recommendation algorithm that had to be done.

All games from App Store are publicly available on iTunes, a media library developed by Apple that lists different kinds of content[3] (e.g. videos, music, games, magazines). When using iTunes library, MARS Agent can find all available games for iOS platform. To find information in such a way that can be easily stored in MARS semantic database, iTunes lookup service by game *id* can be used. When opening game description in iTunes library, *id* of the game is written in the URL. For example, for Candy Crush Saga game, the lookup request is: http://itunes.apple.com/lookup?id=553834731. One of the attributes that is necessary for MARS functioning is genres, and it lists all genres under which game can be categorized. Some of the other available attributes are: screenshot URLs, game description, minimum iOS version supported, etc.

On Google Play users can find all games officially available for Android devices. However, there is no service similar to lookup service on iTunes which can be used to obtain all information about games in such a way that can be easily stored. Google Play is optimized to provide an insight into applications of a registered user, and recommends new games that are compatible with user devices. However, since information about games available on this store is considered important, there exists an unofficial API[4] for extracting information about games available on Google Play. This API is available as a Java library and can be used in every Java application. Some of the available attributes are game genres, average user rating, user rating count, file size, etc.

Since we want to analyze game data, it was necessary to store it into MARS semantic database. Some of the attributes we consider important and necessary for our system functioning are: author ID, author name, price, currency, genre, name, average rating, rating count, game ID, description, and downloads count - necessary for calculating game similarity; supported devices for determining compatibility with users' device; and screenshots for showing how the game looks like. Since we use a semantic database, we also stored data that is specific for a certain platform, e.g. minimum iOS version supported fetched from App Store or a minimum Android version supported fetched from Google Play.

[3] List of the games available on App Store is available at the following link: `https://itunes.apple.com/us/genre/ios-games/id6014?mt=8`

[4] Unofficial API for downloading information about application on Google Play is available at: `https://code.google.com/p/android-market-api/`

4.2 Collecting Inputs Inside the Games

Inside the games we collect information about user experience that is in correlation with user involvement in games: time spent in playing each level of the game, the result of playing a certain level (weather it was successfully completed or not), and total time spent in playing the game. Monitoring of the aforementioned parameters can be easily implemented into a new developed game, like the one mentioned in [3]. However, implementing monitoring into already existing games proved to be quite a challenging task.

The easiest way would be to monitor game parameters from another application. However, on iOS it is impossible to monitor an application from external one since each application has its *sandbox*[5] in which it stores its local settings and data that are not accessible from other applications. On the other hand, this approach is possible to be implemented on Android OS by registering events necessary for user monitoring into a log that is then analyzed by User Agent. However, the problem with this approach is that logging has to be implemented into all games. This is possible for open source games, but the downside of this approach is that monitoring could not be generalized for every game because every game is unique in a sense that different methods are called to create new levels, or that user level results cannot be captured uniformly for all games.

Since we concluded that monitoring of existing games would be very demanding and almost impossible to achieve after the game is developed, we wanted to explore if monitoring could be implemented during the development of the game. Games can be developed by using Android or iOS API, but such games are usually very limited, with limited graphic options and playability. To enhance user experience by providing more powerful visual features and making games more interesting to users, game engines and frameworks can be used [14]. They simplify game development process because they contain various functions which can be used for scene construction, animations, etc. One of the most popular game engines is Unity3D[6] that is available for both Android and iOS platforms. This engine is commercial and its usage is charged, but there also frameworks like Corona SDK[7] and libGDX[8] that are free.

Our idea was to modify game frameworks by adding User Agent that would send event information (e.g. time at which some level was started/finished) to MARS Agent. We used specialized frameworks for each platform: Andengine[9] and Cocos2D[10]. However, this approach also had aggravating factors. For example, the same method can be called when starting a new level or when only

[5] Sandbox is an environment in which every application has limited access to data and settings within OS, network resources, etc.
[6] Unity3D game engine is available at: `http://unity3d.com`
[7] Corona SDK framework is available at:
`http://coronalabs.com/products/corona-sdk`
[8] libGDX framework is available at: `http://libgdx.badlogicgames.com`
[9] Andengine for Android available at: `http://andengine.org`
[10] Cocos2D for iOS available at: `http://www.cocos2d-iphone.org`

changing the scenario within the same level. Thus, we concluded that we cannot implement User Agent functionalities into the existing game frameworks.

This problem could be solved if there was a way to officially define and standardize that a certain method has to be called when a new level is started or finished. However, in order to achieve that, the consent of framework developers and users would be needed. By trying out different methods, we concluded that currently it is impossible to implement User Agent functionalities in existing game engines or frameworks to enable efficient gameplay monitoring. Nevertheless, it is possible to implement our model in the existing games by integrating User Agent functionalities within the game code. In our case we choose Logic game for Android and Climbers game for iOS. In these games, User Agent is implemented as a separate class that consists of five main methods: *levelStarted*, *levelCompleted*, *levelFailed*, *gameStarted* and *gameFinished*. Each of these methods calls the *sendRequest* method that sends a message containing a timestamp of the occurred event to MARS Agent. Additionally, in the message that is sent when level is finished (called by *levelCompleted* and *levelFailed* methods), information whether the level was successfully completed or not is attached.

5 Adaptation of MARS Recommendation Algorithm

MARS Agent collects information about games and events during gameplay for each user and calculates time spent within each level and within the whole game. This information is stored in semantic database on server and used to form user and game profiles. User profiles store information about levels of motivation, progress and success for each user and each game. We use three levels for each parameter: not motivated, averagely motivated and highly motivated; not making progress, making progress and no more room for progress; and successful below the average, within the average or above the average. For instance, user x who plays game y can be seen as averagely motivated, making progress and successful within the average. These parameters are calculated based on data sent by User Agent which are compared to historic data about that game for that user, and historic data about that game for other users. Parameters included in game profiles are filtered from information obtained from Google Play and App Store: genre, game author, description, price and rating. Since the game difficulty parameter, previously proposed in [2], could not be obtained, recommendation algorithm had to be adapted and user and game profiles enhanced in order to be able to function on a wider number of games on mobile platforms.

User profiles store information about user levels of motivation, progress and success for each game she/he played and they are grouped in such a way that for each game genre, user experiences are aggregated so that they present user average motivation, progress and success levels for that genre. User profiles that have the same user experience values are grouped together. On the other hand, games are grouped based on similarities of information stored in their profiles: genre, game author, description, price and rating. For each game, similarity factors with other games are calculated. If two games belong to the same genre, have the same author or rating, and have similar price and words in their description, they are going to have a larger similarity factor than games of different genres/authors/ratings and totally different set of words in descriptions and prices.

Our adapted version of the algorithm uses a hybrid approach to recommend games because it includes both user profiles grouping (i.e. collaborative approach) and game profiles grouping (i.e. content-based approach). Besides user and game profile groups, an input for the algorithm is also user motivation for the game she/he is currently playing. The algorithm then makes a list of games, which the user did not play before, that were played the most by users belonging to the same user profile group. Moreover, the algorithm also makes a second list of games. If user is considered averagely of highly motivated for playing the current game, a game from the same game profile group is put into the list. On the other hand, if she/he is not considered motivated for playing this game, random games from different game profile groups are put into the list.

The algorithm then combines two aforementioned lists into one list from which it randomly chooses k games. Our recommendation list uses information collected from both platforms which is, to the best of our knowledge, a novel approach because currently game recommendations are performed only within a same platform. The produced game list is then filtered so that it contains only games available on user smartphone. Those games are then sent to the User Agent and displayed in the advertisement area of the currently played game.

6 Conclusion

In this paper we presented steps we had to take to enable implementation of Multi-Agent Recommendation System for Games on Mobile Phones (MARS) on most popular mobile platforms - iOS and Android. To integrate MARS system with aforementioned platforms, two types of agents needed to be implemented - User Agent that monitors user experience and MARS Agent that collects data about games, analyses user experience data obtained from User Agent, and creates game recommendation lists.

The most challenging aspect of MARS implementation process was to enable gameplay monitoring. We explored different ways of external application monitoring and came to conclusion that it was not possible to enable logging of interesting events within games without going into the game code. Furthermore, we also tried to implement gameplay monitoring in different frameworks used for game development and concluded that this approach was also not possible without standardization of methods which are to be used for particular events.

Therefore, we implemented MARS gameplay monitoring into only two open-source games: Logic game for Android and Climbers game for iOS. Since one of the parameters important for previous version of MARS algorithm was not possible to be obtained - difficulty of the game, we had to improve the existing algorithm. Another contribution of this paper is the definition of user and game profiles that enable calculating similarities between users and games and which are used as inputs for improved recommendation algorithm. For future work, we plan to implement the adapted version of MARS algorithm with newly defined profiles. After the implementation, the influence of our approach on user satisfaction and downloading recommended games will be tested.

Acknowledgements. We would like to acknowledge support of our students: Marin Glibic, Mladen Kubat, Marko Milos, Marko Pavelic and Stjepko Zrncic.

References

1. Pripuzic, K., Zarko, I., Podobnik, V., Lovrek, I., Cavka, M., Petkovic, I., Stulic, P., Gojceta, M.: Building an IPTV VoD Recommender System: An Experience Report. In: Proceedings of the 12th International Conference on Telecommunications, pp. 155–162 (2013)
2. Skocir, P., Marusic, L., Marusic, M., Petric, A.: The MARS – A Multi-Agent Recommendation System for Games on Mobile Phones. In: Jezic, G., Kusek, M., Nguyen, N.-T., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2012. LNCS, vol. 7327, pp. 104–113. Springer, Heidelberg (2012)
3. Skocir, P., Jezic, G.: A Multi-Agent System for Games Trading on B2B Market Based on Users Skills and Preferences. In: Advanced Methods and Technologies for Agent and Multi-Agent Systems, pp. 303–312 (2013)
4. Oh, J., Kim, D., Lee, U., Lee, J.G., Song, J.: Facilitating Developer-user Interactions with Mobile App Review Digests. In: Proceedings of Extended Abstracts on Human Factors in Computing Systems, pp. 1809–1814 (2013)
5. Medler, B.: Using Recommendation Systems to Adapt Gameplay. In: Discoveries in Gaming and Computer-Mediated Simulations: New Interdisciplinary Applications, pp. 64–77 (2011)
6. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering 17, 734–749 (2005)
7. Roh, J.H., Jin, S.: Personalized Advertisement Recommendation System Based on User Profile in the Smart Phone. In: Proceedings of the 14th International Conference on Advanced Communication Technology, pp. 1300–1303 (2012)
8. Jannach, D., Hegelich, K.: A Case Study on the Effectiveness of Recommendations in the Mobile Internet. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 205–208 (2009)
9. Costa-Montenegro, E., Barragans-Martinez, A., Rey-Lopez, M., Mikic-Fonte, F., Peleteiro-Ramallo, A.: Which App? A recommender System of Applications in Markets by Monitoring Users' Interaction. In: Proceedings of the International Conference on Consumer Electronics, pp. 353–354 (2011)
10. Sorensen, D.R.: Recommending Games on Twitter. Master's thesis, The Royal School of Library and Information Science, Copenhagen, Denmark (2013)
11. Shin, C., Hong, J.H., Dey, A.K.: Understanding and Prediction of Mobile Application Usage for Smart Phones. In: Proceedings of the ACM Conference on Ubiquitous Computing, pp. 173–182 (2012)
12. Böhmer, M., Hecht, B., Schöning, J., Krüger, A., Bauer, G.: Falling Asleep with Angry Birds, Facebook and Kindle: A Large Scale Study on Mobile Application Usage. In: Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 47–56 (2011)
13. Yang, H.T., Chen, D.Y., Hong, Y.X., Chen, K.T.: Mobile Game Recommendation Using Touch Gestures. In: Proceedings of NetGames, pp. 1–6 (2013)
14. Gregory, J.: Game Engine Architecture. Taylor & Francis (2009)