AppTrends: A Graph-based Mobile App Recommendation System Using Usage History

Donghwan Bae, Keejun Han, Juneyoung Park, Mun. Y. Yi
Department of Knowledge Service Engineering
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
Email: {bdhwan, keejun.han, j.park89, munyi}@kaist.ac.kr

***Abstract*—With the advent of smartphones, mobile phones have evolved from a simple communication tool to a multi-purpose device that affects every aspect of our daily life. The expansion of the mobile application market has made it difficult for smartphone users to find applications that fit their needs. Most prior research on application recommendation provides a limited solution to the problem of application overload. These recommendation techniques, developed outside of the mobile environment, have a number of limitations such as cold start problem and domain disparity. In this paper, we propose AppTrends, which incorporates a graph-based technique for application recommendation in the Android OS environment. Our experiment results obtained from the field usage record of over 4 million applications clearly show that the proposed graph-based recommendation model is more accurate than the Slope One Model.**

Keywords— Mobile application recommendation; Pathfinder network algorithm; Usage graph; Smartphone

I. INTRODUCTION

The rapid emergence of smartphones has inevitably formed a new online market for mobile application. According to a recent study, at least 10 billion people use smartphones in their daily life for various purposes varying from a simple phone call to delicate activities such as document editing [1]. Among those various purposes, approximately 66% of the users mainly use their smartphone to experience new mobile applications, closely followed by Internet surfing [2]. This statistic tells us that the mobile application plays a vital role in people's daily activities.

The mobile application can be easily installed from an application market such as Apple's App Store and Google's Android Market. The simple process of accessing the market for both providers and downloaders have allowed the Android market to grow and to register 0.7 million applications and is continously increasing. [3].

Whether the user has installed an application, however, is a vague indicator of whether the user actually likes the application. In accordance with the recent survey [3], nearly 3 out of 10 applications are instantly deleted after installation. This finding implies that finding the exact application one requires can be a difficult process. This difficulty also poses harm to the developers, as they lose the chance to promote their applications to the users because of the enormous number of applications that already exists in the market. Another survey [4] presents that the top 50 applications reach almost 60% of total download number; and the rest 0.69 million applications are fighting for the rest 40%, indicating that most of the applications fail to even be discovered by the users. The nature of the market creates a definite need for a recommendation technique specialized for mobile applications.

One of the most reliable recommendation technique for mobile applications is based on Collaborative Filtering (CF) method. Collaborative Filtering generates recommendations primarily using users' ratings. Although it is widely adapted and promises a reliable performance, it suffers from cold start problem, which is a critical issue when the user-generated data is insufficient. Especially, in recommending mobile applications, a cold start happens more frequently as the life span of the mobile application is very short for sufficient collection of rating data from the user [5].

Figure 1 A sequential list of the applications for a specific task (i.e., uploading a photo to SNS and finding a restaurant nearby)

To counteract the application overload problem, in this paper we present AppTrends, a recommendation system for mobile applications. The novel feature of AppTrends is that it generates a graph from a user's app usage history. In the application-based graph, the nodes are the applications used by the user and those nodes are connected if they are used simultaneously in a certain session. The simultaneous use of application implies that those two connected applications have a high chance of being co-used to complete a sequential task. For instance, Figure 1 shows the applications used to upload a photo to a social network service. The process requires a camera application to take the photo, which is then edited through a photo editing application and shared via a social network application. Although a different combination of applications can be chosen to complete an identical task, it sounds plausible to consider those applications are co-related

once they are used together. A newly-generated graph, using the co-related applications, is then used to make a personalized recommendation for mobile applications. Unlike the other existing solutions, AppTrends is completely automatic and is flexible to deal with the cold start problem. To the best of our knowledge, AppTrends is the first mobile application that generates a graph-based recommendation for mobile applications.

The major contributions of this paper are as follows:

- We propose a novel recommendation technique for mobile applications, which is based on co-occurrence of mobile applications activated in sequence. We then compare our model with the CF method, a fundamental model of a current state-of-the-art commercial model, The results of our experiment show that AppTrends outperforms the baseline model.
- We collect a real usage data from users for our experiment in order to validate whether AppTrends delivers a practically meaningful results for the recommendation of the mobile application. To do so, we have implemented the mobile prototype of AppTrends and collected real usage data during two weeks from 206 users.

The remainder of this paper is structured as follows: In section II, we review the existing techniques used in mobile application recommendation. In section III, we propose our graph-based recommendation system, called AppTrends. In section IV, we demonstrate the effectiveness of our system through an experiment and present the results. In section V, we conclude our paper.

II. Background and Related Work

In this section, we discuss some of the conventional recommendation techniques that are available for mobile application and introduce some of the real mobile services that are similar to our system.

A. Recommender Systems in E-commerce

A general strategy for item recommendation in E-commerce is to discover potential items that are liked by specific users. There are mainly four types of recommendation strategies: Non-personalized, Attribute-based, Item-to-Item Correlation, and People-to-People Correlation.

First, Non-personalized recommendation technique focuses on discovering items that have relatively higher rating scores, mainly targeting to satisfy most users. In other words, this technique does not regard each user's interest, but rather recommends the same items that are sold most with high rating scores to all users [5]. Second, Attribute-based recommendation technique returns categorized items by analyzing user's specific attributes. Those attribute can be collected manually from user input or automatically collected from exploiting user's usage data. A service based on this technique is Movie Map [6], which recommends movies from a category where a user is interested in. Third, Item-to-item correlation is to recommend items that are not seen by a user, but similar to items that the user has purchased. With the purchasing history of the user, this technique can be built in an automatic way. Amazon [7] is one of the most popular online shopping services that adopts this technique for their item recommendation. Lastly, People-to-People recommendation measures how similar a user is to other users and recommends items that are liked by the users who are similar to the user. The commercial services based on this technique are Netflix [5] and Musicstrands [9] that recommends songs with regards to various features of the song such as genre, album, and singer.

B. Recommender Systems for Mobile Application

Recommender systems are not only used in the E-commerce market environment, but also in the mobile environment to recommend potentially favorable mobile applications for smart phone users. Upon the analysis for commercial mobile application recommender systems, we classify them into following two categories: Social-aspect recommender system and context aware recommender system.

Social-aspect recommender system is a technique that discovers new items by sharing user information with others. One of the mobile application recommender systems, AppAware [10], records installation, update, deletion of mobile applications and share those records with others, in order to make a recommendation. This system is built upon the assumption that word-of-mouth is the most effective way to expand the influence of mobile application [11]. However, the limitation of this system is that it suffers from the cold start problem for new users as they have not created enough number of user data to be analyzed for recommendation. Furthermore, finding similar groups of users is not an easy task without acquiring a large amount of rating scores for mobile applications. Unlike other items in E-commerce, mobile application market is still an emerging market, meaning that most of mobile application misses the votes and ratings by users. With those data missing, the recommendation would fail to satisfy the user's needs. However, our system, AppTrends, uses the user's own application usage data, so it would alleviate the problem caused from data sparseness .

On the other hand, context aware recommender systems attempt to understand user's current situation and reflect those information into recommendation. The possible components of the context can be time, location, activity, weather, emotional status, and social status [12]. Understanding those contexts enables the recommender systems to be more adaptable depending on user contexts. For instance, a user currently on a bus going for a train station is highly likely to be interested in a mobile application that notifies her subway schedule. Appazzar [13] recommends mobile applications by considering the user's current physical location. AppAware [14] is also a location-based recommendation because it provides the information of mobile application download done nearby to the user's current location. Inspired by AppAware, Applause [15] was further developed to overcome the cold start problem by asking their favorable places from newbies so as to recommend mobile application downloaded within those places.

Although the above naive approaches of understanding user's context are promising, it is yet unsatisfactory to understand what the user is currently doing. User's current activity can be instantly associated with the user's current interest, implying that those contextual information can be

exploited to improve the performance of the recommender systems. Unlike the recommender systems above, our proposed recommender system, AppTrends, employs a novel approach that focuses on co-occurrence of mobile applications while conducting a sequential task. By doing so, AppTrends is expected to provide more adaptable recommendation results for each user by capturing the user's current interest (i.e., taking a photo, finding a route) from the co-occurrence information.

III. Our proposed system

In this section, we first introduce an automated method for generating a mobile-application based graph from their co-occurrence information, then moving on to explain how to implement our system, AppTrends.

A. Algorithm for AppTrends

The algorithm for AppTrends largely consists of two steps: Mobile application-based graph creation and Fit-score calculation. In the subsequent sections, we explain each step in detail and how each step generates the final recommendation result for mobile application.

1) Mobile Application-based Graph Creation:

The proposed approach that automatically generates mobile application-based graph from a user's application usage data requires two specific information of the soruce: A set of applications and co-occurrence scores of the applications. These pieces of information are then processed with a number of refining steps to remove weak connections for noise deduction by calculating the shortest distance between the applications.

As a first step, the mobile application list of a user are created from the user's mobile application usage history. To caputre the applications, AppTrends records a list of the mobile applications of the user. Once the user downloads an application, the application is registered to AppTrends.

The co-occurrence between the applications are measured during a session. A session begins when the user turns on the mobile phone and terminates when the screen goes off. The applications used within this session is considered to have a co-occurrence relations. For instance, if a user wants to take a photo and share it with her friend, the user turns on her mobile, indicating that a new session begins. Then she takes a photo by turning on the camera application. After editing the photo, she sends the photo to her friends by using a messenger application. Lastly, she turns off the application, at which the session ends. Within this session, the user used camera, performed photo editing and used messenger application, in sequence, to complete a specific task.

As these applications are used to achieve the same purpose, we can define that these applications are in a co-occurrence relation. As such cases happen more frequently, the distance between two applications in the graph becomes shorter. The session co-occurrence, denoted as SC, between two application A_i and A_j in the user u , can be formally defined as follows:

$$SC(A_i, A_j) = \sum_{k=1}^{n} n(A_i \cap A_j) \quad (1)$$

$$distance_{sc}(A_i, A_j) = \frac{SC(A_i, A_j)}{\max(SC)} \quad (2)$$

where n is the number of sessions, $n(A_i \cap A_j)$ is the number of session co-occurrence of two application A_i and A_j, max(SC) indicates the maximum SC between any applications for the user u, for normalization. As two applications more often occur in the same session, the distance between the applications need to be closer, thus, defining the distance between A_i and A_j can be obtained as follows:

$$distance(A_i, A_j) = 1 - distance_{sc}(A_i, A_j) \quad (3)$$

Lastly, to reduce the noise between applications, we used Pathfinder network pruning [16] that calculates the minimum distance between nodes. The underlying priciple for calculating minimum distance between two nodes N_a and N_e is Triangle Inequality [17]. Figure 2 shows the example of calculting the minimun distance between N_a and N_e.

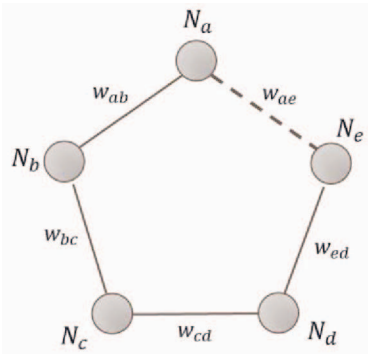

Figure 2. An illustration of generating edge based on triangle inequality

In this case, the edge w_{ae} connecting N_a and N_e is only drawn as follows:

$$w_{ae} \leq (w_{ab}^r + w_{bc}^r + w_{cd}^r \cdots + w_{de}^r)^{\frac{1}{r}} \quad (4)$$

where r is a parameter to balance the weight of neightbor distances and fixed at 1 for simplicity in this paper.

Through the step (1) to (4), we now can generate the mobile application graph for the user u where a node is an application installed on the mobile phone and an edge is the connection between two application that are used during a certain session, Figure 3 illustrates an example of such a graph. The size of the node indicates that the frequency of use of the application while the thickness of the edge represents the distance between the two applications. The thickness of the edge represents the frequency of co-occurrence between the two applications.

2) Fit Score: A method for Recommending an Application

Once the graph is created, we now can recommend an application for the user by calculating a fit score. Fit score represents the predicted score of an edge for new node when the node is newly connected to the graph. In other words, it means the probability score whether the application can be

connected to the graph. This score increases when the graph is similar to another graph where the new application already exists and the distance from the application to other applications is closer. The score also increases if the user who has a similar graph frequently uses the application.

Figure 3. An example of the application-based graph

To do so, we need to measure the similarity between the graphs via Minimum Common Subgraph (mcs) [18] measure. The similarity between two graph increases when they have plenty of common nodes and edges connecting to the nodes. The graph similarity, GS, between G′and G″ can be measured as follows:

$$\mathrm{GS}(G', G'') = \frac{|\mathrm{mcs}(G', G'')|}{\max(|G'|, |G''|)} \quad (5)$$

where $|\mathrm{mcs}(G', G'')|$ indicates the number of common edges between G′ and G″ and $\max(|G'|, |G''|)$ is the maximum number of nodes.

Figure 4 illustrates an example of measuring the similarity of two graphs to represent user A and B, respectively. There are three common edges, denoted as $|\mathrm{mcs}(G', G'')|$, and the maximum number of nodes, denoted as $\max(|G'|, |G''|)$, is 8 from the user B. Therefore, the similarity between the two graphs is 0.375.

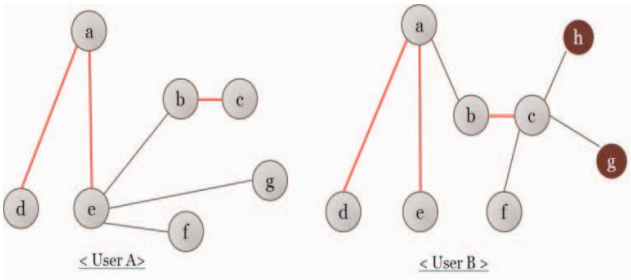

Figure 4. An example of measuring graph similarity

To predict the fit score of new application for the user u, AppTrends requires the user's graph, G_u and other user's graph, G_u'. Chosen a random node n in G_u, it then finds a potentially undetected application for the user by detecting the neighbor nodes that are already connected to the chosen node n from G_u'. By doing so, it becomes possible to calculate a probability of how much the user likes the discovered application n′ by multiplying the edge distance of the node and the similarity score between two graphs as follows:

$$F(G_u, G_u', n') = GS(G_u, G_u') * \frac{1}{distance(n, n')} \quad (6)$$

Based upon the above approach for calculating the fit score, AppTrends calculates the fit score for all the potential nodes that can be added to the nodes in the graph G_u. Then it returns those applications for the user u in a descending order of the fit score.

Figure 5 UI for AppTrends

B. *Implementation for AppTrends*

To verify our proposed algorithm for AppTrends, we developed a prototype of AppTrends. It was developed as an Android application. The application largely contains four functions: Usage collector, Personalized application recommendation, App usage graph, and Friend app usage graph browsing.

Usage collector collects the installation and usage history of mobile applications for a user and sends it to the AppTrends server. Other functions interact with the user throughout the user interface shown in Figure 5. Personalized application recommendation function shows a list of applications that are recommended for the user. Each item in the list consists of icon, name, developer, description, fit score, and rating score of the application. App usage graph function visualizes a usage graph of the user from her usage data. On the other hand, Friend app usage graph function visualizes the user's friends on Facebook. By choosing a friend's name on the list, AppTrends shows the chosen friend's usage graph. In addition, AppTrends supports another prototype function, called World App Trends Map, that

shows a global mobile application graph by analyzing all the users' usage history. As it exploits all the usage data, it enables a user to visualize the all trends of the mobile applications.

IV. EXPERIMENTS

In this chapter, we will first introduce our dataset, compared method and evaluation metrics that are used for our experiment. Then, we investigate the performance of our proposed algorithm by comparing it with the current state-of-the-art mobile application recommender algorithm. We also introduce the graph integrating all users' usage patterns that can be used for making a recommendation, especially for new users.

A. Data Set

To evaluate our algorithm, we collected the application usage data from users who installed AppTrends on their mobile. For a precise analysis, we only chose 206 users who have used AppTrends more than two weeks. These users made 4,201,041 applications usage data and 713,745 sessions in total. Some applications such as launcher and screen lock application are removed from the dataset in order to decrease the noise effect. The usage data during the first week is used for training, and the other during the second week is used for testing.

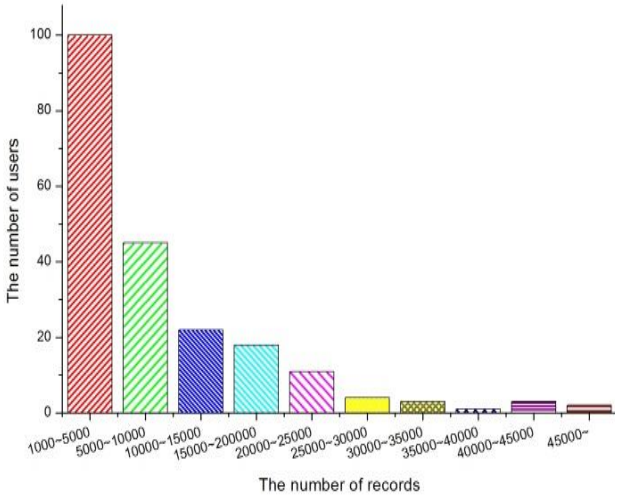

Figure 6. The number of records for individual users

Figure 6 shows a proportion of users with respect to their number of records. Most of users used the applications on mobile from 1,000 to 5,000 while 18 users use the application more than 15,000, indicating that they are heavy mobile users. Furthermore, Figure 7 shows which category is more used by general smartphone users in application market. It seems that the applications categorized in Life Style, Tools, and Educations are much more used compared to applications in the other categories. Similarly, Figure 8 lists the most used mobile applications by the users, showing that Social Network Service applications such as KakaoTalk [19] and Facebook [20]. Note that the X axis in Figure 8 is a log scale, indicating that KakaoTalk is a dominant application among the applications available to users.

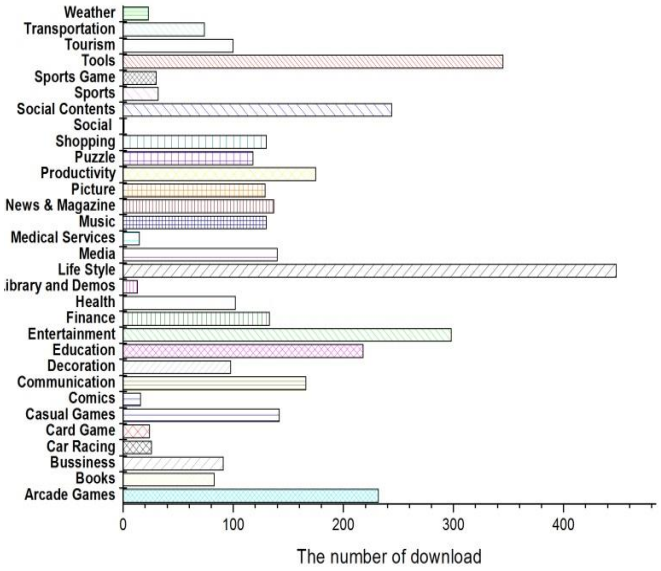

Figure 7. The number of the applications per category

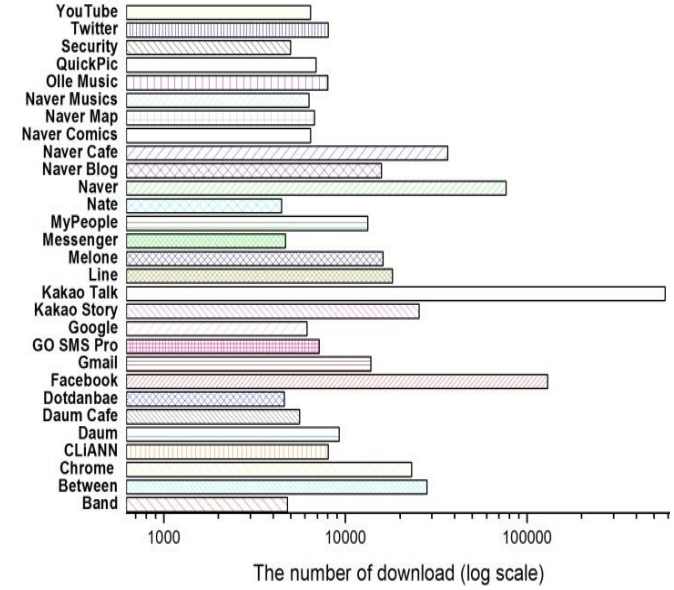

Figure 8. Top 15 popular applications detected by AppTrends

B. Compared Methods and Evaluation Metric

To evaluate the performance of AppTrends, we compare it with one of the state-of-the-art recommender algorithms for the mobile application. Slope One predictors [21], takes into account information from other users who rated the same item and from the other items rated by the same user. The ratings from a specific user is represented as an array u including training and testing set, where u_i is the rating of this user gives to item i. The subset of the set of items consisting of all those items which are rated in u is denoted as S(u). Then the average deviation of item i with respect to item j is as follows:

$$dev_{j,i} = \sum_{u \in S_{j,i}(R)} \frac{u_j - u_i}{|S_{j,i}(R)|} \quad (7)$$

where R is a training set within u , any two items i and j with ratings u_i and u_j respectively in some user evaluation u ,

denoted as $u \in S_{j,i}(R)$, and $|S_{j,i}(R)|$ is the number of elements in a set S. Given that $dev_{j,i}$, the predictor for the Slope One is defined as follows:

$$pred(u,j) = \frac{\sum_{i \in Releevant(u,j)}(dev_{j,i} + u_i)}{|Relevant(u,j)|} \quad (8)$$

where $|\mathrm{Relevant}(u,j)|$ is the number of all relevant items in $S(R)$.

As Slope One algorithm outperforms other recommender algorithms for online rating-based collaborating filtering [21], except for variants of the Slope One, we compare AppTrends with the Slope One in order to evaluate the effectiveness of our model.

Furthermore, to evaluate our model, Precision @N is used. The number of k recommended applications ($k \leq N$) , in descending order of prediction score, are evaluated by counting how many applications actually exist in the user's testing dataset.

C. Experiment Results

Figure 9 shows the comparison between our model, AppTrends, and the compared model, Slope One, with different N (i.e., size of recommendation result set) on Precision @N measure for our dataset. From Figure 9, we can see that our model outperforms the compared model for all values of N, except for $N = 10$. The higher value of N is, the higher value of Precision AppTrends gets, indicating that our model can retrieve more relevant applications for the given user. In overall, 60% improvement is achieved by adapting AppTrends compared to Slope One.

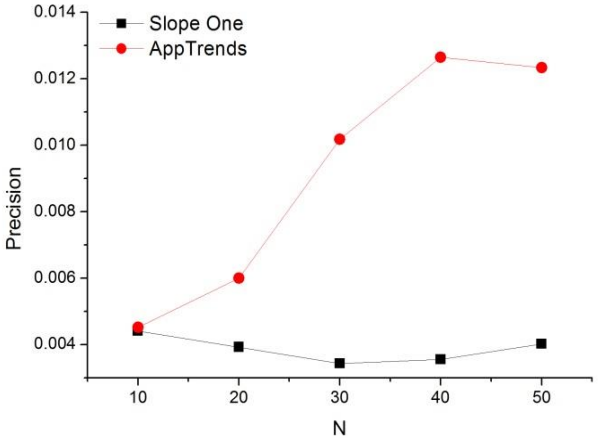

Figure 9. Precision scores with various N for Slope One and AppTrends

D. Visualization

Finally, we can visualize all users' application usage data as a graph representing a usage pattern of the all mobile applications. The generated graph is useful, especially for new users who do not have enough mobile application usage data for analysis. In a conventional method, it requires a considerable effort and time to alleviate the cold start problem. On the other hand, compared with the conventional recommendation models, our graph-based approach is relatively faster to generate a list of recommended applications without a series of complex computing process for recommendation. Once the graph is created, the recommendation for new users can be done offline as the system can instantly choose neighbor applications of what the user installed on her mobile. This implies that our model can be more effective when the system is targeted for a commercial service in practice. Figure 10 shows the graph to visualize all users' application usage patterns.

Figure 10. An illustration of mobile application graph for all users

V. Conclusion and Future Works

The purpose of this paper is to develop a recommendation model specialized in the mobile applications on smart phones. To do so, we have introduced a novel method for representing a user's usage pattern as a graph where nodes are applications installed by the user and edges are distances of how close two connected applications are. Based upon the graph, graph similarity and fit score are measured to predict the recommendation score for unseen applications to the given user. Throughout our experiment, we showed that our model, AppTrends, clearly outperforms the compared method. Furthermore, our graph-based approach creates a global mobile-application graph representing a global application usage pattern, which can be utilized for addressing data sparseness issues for new users.

Taking a hybrid approach and combining the proposed recommendation model of AppTrends with other state-of-the-art recommender models may further improve the recommendation performance. Recommender models are actively investigated in recent years and adaptation of the several state-of-the-art models has been proposed. Thus, an attempt to develop a hybrid system of AppTrends with other models is expected to improve the overall performance of recommendation results.

Acknowledgement

We thank the editor and anonymous reviewers for their helpful feedback on the earlier versions of this paper. This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF-2011-0024560).

REFERENCES

[1] http://www.businesswire.com/news/home/20121017005479/en/Strategy-Analytics-Worldwide-Smartphone-Population-Tops-1/.

[2] http://isis.kisa.or.kr/board/?pageId=060200&bbsId=3&itemId=799/.

[3] http://www.appbrain.com/stats/number-of-android-apps/.

[4] http://appsfire.com/infographics/apps_vs_webapps_150dpi.png/.

[5] J. Bennett and S. Lanning, "The Netflix Prize," 2007.

[6] Reel Cinemas, http://www.reelcinemas.ae/Movies/.

[7] Amazon, http://www.amazon.com/.

[8] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in E-commerce," pp. 158-166, 1999.

[9] G. Holmberg, "Musicstrands TM: A platform for discovering and exploring music, 2005.

[10] A. Girardello and F. Michahelles, "Bootstrapping your mobile application on a social market," In *Proc. of UbiComp*, 2010.

[11] Z. Ahmet, and K. V. Mattila, "Mobile service distribution from the end-user perspective – The survey study on recommendation practices," In *Proc. of CHI Extanded Abstract*, pp. 573-588, 2012.

[12] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Journal of Human-Computer Interaction*, 16(2): 97-166, 2001.

[13] M. Bohmer, G. Bauer, and A. Krugger, "Exploring the design space of recommender systems that suggest mobile apps," In *Proc. of workshop CARS*, 2010.

[14] A. Girardello, F. Michahelles, "AppAware: which mobile applications are hot?," In *Proc. of MobileHCI*, pp. 431-434, 2010.

[15] C. Davidsson, and S. Moritz, "Utilzing implicit feedback and context to recommend mobile applications from first use," In *Proc. of workshop CaRR*, pp. 19-22, 2011.

[16] S. Hauguel, C. Zhai, and J. Han, "Parrel pathfinder algorithms for mining structures from graphs," In *Proc. of ICDM*, pp. 812-817, 2009.

[17] A. Tversky, and I. Gati, "Similarity, separability, and the triangle inequality," *Psychological Review*, 89:123-154, 1982.

[18] H. Bunke, X. Jiang, and A. Kandel, "On the minimum common supergraph of two graph," *Computing*, 65(1):13-25, 2000.

[19] Kakaotalk, http://www.kakao.com/talk/.

[20] Facebook, http://www.facebook.com/.

[21] D. Lemire, and A. Maclachlan, "Slope one predictors for online rating-based collborative filtering," In *Proc. of SDM*, pp. 471-475, 2005.