Serendipitous Recommendation for Mobile Apps Using Item-Item Similarity Graph

Upasna Bhandari[1], Kazunari Sugiyama[1], Anindya Datta[1], and Rajni Jindal[2]

[1] School of Computing, National University of Singapore,
Computing 1, 13 Computing Drive, Singapore 117417
[2] Department of Computer Science, Delhi Technological University,
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India
{a0106246,sugiyama,datta}@comp.nus.edu.sg, rajnijindal@dce.ac.in

Abstract. Recommender systems can provide users with relevant items based on each user's preferences. However, in the domain of mobile applications (apps), existing recommender systems merely recommend apps that users have experienced (rated, commented, or downloaded) since this type of information indicates each user's preference for the apps. Unfortunately, this prunes the apps which are releavnt but are not featured in the recommendation lists since users have never experienced them. Motivated by this phenomenon, our work proposes a method for recommending serendipitous apps using graph-based techniques. Our approach can recommend apps even if users do not specify their preferences. In addition, our approach can discover apps that are highly diverse. Experimental results show that our approach can recommend highly novel apps and reduce over-personalization in a recommendation list.

1 Introduction

Concurrent with the phenomenal spread of smart-devices (*e.g.*, iPhone, iPad), the mobile application (app) market has experienced explosive growth. For instance, Apple's iOS App Store offers more than 550,000 unique apps to users in 123 countries, along with download counts exceeding 25 billion[1]. The enormous scale of the app market makes it difficult for users to discover apps that are relevant to their interests. In this context, it is tempting to apply recommender systems (RSs), which have been used successfully in a variety of domains like movies and books to alleviate the problem of information overload by suggesting items directly to users relevant to their interests.

In general, the effectiveness of RSs has been shown to be proportional to the data sparsity of the underlying application domain. In other words, the larger the fraction of the item corpus that has been experienced by users, the better the quality and coverage of recommendations from that corpus. Effectively, if an item has not been experienced at all, the RSs is not able to recommend items similar to it. This feature of RSs create a stiff hurdle for them to be applied in mobile app recommendations, as the rate of introduction of mobile apps is extraordinarily high, and the fraction of apps that have been experienced (downloaded) by users is extremely low.

[1] http://www.apple.com/pr/library/2012/03/05Apples-App-Store-Downloads-Top-25-Billion.html

R.E. Banchs et al. (Eds.): AIRS 2013, LNCS 8281, pp. 440–451, 2013.

The most popular approach in recommender systems is collaborative filtering (CF) [11,17,13] that works by recommending items to target users based on what other similar users have previously preferred. Another popular technique is content-based filtering (CBF) [7,6,21] that provides recommendations by comparing representations of content contained in an item with representations of content that the user is interested in. Both CF and CBF, however, are mostly aimed at generating accurate recommendations that is relevant to user's interests. The lack of "surprise" element in these recommendations owing to the fact that there are ratings available for only a small fraction of apps creates a major hurdle in overall user satisfaction of the user.

Recently, some researchers have focused on developing serendipitous recommendation systems [5], [12], [14], [16], [22], [1]. It is reasonable to say that a user would be happy with recommendation systems that offer less obvious choices. Suppose that we visit Amazon.com[2] to buy something online. To illustrate, after browsing a couple of items, Amazon.com provides us with lists such as *Recommended For You* or *Customer Who Bought This Item Also Bought*. Looking at them closely, users often observe that all of these items are already known. This may not be ideal for overall user satisfaction and experience with the system. For example, if a user browses a book written by *Dan Brown*, most of the recommendations for the user will be books by *Dan Brown*.

Serendipitous systems work on the basis of the assumption that the user may want to be surprised with something unexpected that he did not start out looking for. E-commerce Web sites, however, usually just offer a long list of search results. Therefore, users have little chance of finding out something different from their preferences. Another problem with existing recommendation systems is that they often recommend items which the users have rated or downloaded before. This limits the candidate apps to be recommended by pruning relevant but not yet rated or downloaded apps.

In order to provide serendipitous recommendations that solve the problems of existing recommendation systems, we first define serendipitous recommendation as the one that provides something diverse and novel, and then we propose a method for providing serendipitous recommendations by increasing item novelty and diversity. We leverage the user's preferences by tapping into the information about apps installed on mobile phones and recommend serendipitous apps using item-item similarity graph.

This paper is organized as follows: In Section 2, we review related work on serendipitous recommendation and state-of-the-art mobile app recommendation systems. In Section 3, we detail our graph-based approach to providing serendipitous recommendation for mobile apps. In Section 4, we present the experimental results for evaluating our proposed approaches and some user analysis. Finally, we conclude the paper with a summary and directions for future work in Section 5.

2 Related Work

In this paper, our goal is to construct a serendipitous recommendation system for mobile apps. Thus, we review related works on serendipitous and mobile apps recommendation systems in the following.

[2] `http://www.amazon.com`

2.1 Serendipitous Recommendation

Most of the recommendation approaches focus on recommending a list of items similar to the items previously seen and rated highly by the target user. However, much fewer works address serendipitous recommendations. Ziegler *et al.* [27] proposed a similarity metric using a taxonomy-based classification and uses it to compute an intra-list similarity to determine the overall diversity of the recommended list. They provide a heuristic algorithm to increase the diversity of the recommendation list. Zhang and Hurley [26] focused on intra-list diversity and optimized the tradeoffs between users' preferences and the diversity of the top-N results. They modeled the competing goals of maximizing the diversity of the searched list while maintaining adequate similarity to the user query as a binary optimization problem. Andre [5] proposed a method for performing serendipitous searches for Web information retrieval. They first defined the potential for serendipity as search results that are interesting but not highly relevant. In another publication, they discussed serendipity from human cognitive point of view [4]. They hypothesized that a reconsideration of serendipity from numerous angles may help refine new opportunities for designing systems to support, if not serendipity exactly, then the desired effects of serendipitous revelation. Lathia [14] found that temporal diversity is an important facet of recommender systems by showing how data of collaborative filtering changes over time. This work has explored temporal aspect of recommendation, but we focus on recommending serendipitous mobile apps to each user. Kawamae [12] emphasized the surprise of each user in the recommendation focusing on the estimated search time that the users would take to find the item by themselves. Their recommender system assumed that items recently purchased by an innovator, who has well-proven unpredictable trait, will surprise other users more than other items. Nakatsuji *et al.* [16] improved the drawback of Ziegler *et al.*'s approach described above. They proposed a method for identifying items that are highly novel for the user by defining item novelty as the smallest distance from the class the user accessed before to the class that includes a target item. Sugiyama and Kan [22] proposed a method for recommending serendipitous scholarly papers. They used the preferences gathered from other users (dissimilar users and co-authors) in the construction of the target researcher's user profile, used in matching candidate documents to achieve serendipitous recommendations. Recently, Adamopoulos and Tuzhilin [1] proposed an approach to providing unexpected recommendations by formalizing the Greek philosopher Heraclitus's concept, "If you do not expect it, you will not find the unexpected, for it is hard to find and difficult."

2.2 Recommendation for Mobile Apps

As a tremendous number of apps are readily available, users have difficulty in identifying apps that are relevant to their interests. Thus, recommender systems for apps have started to gain popularity. To understand how, where, and when apps are used compared to traditional Web services, Xu *et al.* [23] investigated the diverse usage behaviors of individual mobile apps using anonymized network measurements from a tier-1 cellular carrier in the United States. Yan and Chen [24] and Costa-Montenegro *et al.* [8] constructed recommendation system for apps by analyzing how the apps are actually used. Mobile devices are usually used in various contexts due to their ubiquitous nature. Davidsson and Moritz [10] developed a prototype system of app recommendation

that achieves such context-awareness by exploiting GPS sensor information. Recommendation for apps needs to consider factors that invoke a user to replace an old app (if the user already has one) with a new app. Focusing on this point, Yin *et al.* [25] introduced the notions of actual value (satisfactory value of the app after the user used it) and tempting value (the estimated satisfactory value if the app seems to bring to the user) and regarded recommendation for mobile apps as a result of the contest between these two values. Based on the observations that app-related information in Twitter can precede formal user ratings in app stores, Lin *et al.* [15] developed a novel approach to recommending mobile apps in cold-start situations.

3 Proposed Method

The serendipitous recommendation works described in Section 2.1 have not addressed apps for smartphones yet. In addition, the recommendation systems for mobile apps described in Section 2.2 have not employed serendipitous recommendation. Thus, to the best of our knowledge, this is the first work on serendipitous recommendation for mobile apps. In this section, we explain our proposed approach.

3.1 Intuition of Our Proposed Method

The state-of-the-art recommendation systems have a key assumption, "*every item must be used at least once and every user must use at least one item.*" According to this assumption, unfortunately, it is highly possible that recommender systems prune items that may be good but never get featured since nobody has used them yet. We mainly focus on increasing the aggregate diversity by discovering highly diverse and new apps to achieve serendipitous recommendations. In our approach, we define new apps as those which have never been downloaded or rated.

Graph-based techniques have been previously employed in recommender systems, but mainly focused on improving accuracy or maximizing diversity [3], [2]. These techniques are popular since they can avoid the problems of sparsity and limited coverage by evaluating the relationships between users or items that are not "directly connected" [18]. Thus, to generate serendipitous recommendations for mobile apps, graph-based methods are useful since they preserve some of the "local" relations in the data. In this section, we propose an approach to generating serendipitous recommendations based on apps installed on a target user's phone by using item-item similarity graph. We believe that, by using the list of apps already installed on a user's phone, we can capture a holistic view of user's preference and not relying on ratings given to a few apps so in turn what we are trying to do is focus on apps that the user has actually downloaded and used. Also, in many cases users download apps to test them out but never install them but we only take into account apps that are installed by the user. The main intuition is that, if there exists a path connecting two apps on a user's phone and the weights on each edge that constitutes this path are above a certain threshold of similarity, apps along this path which are not already downloaded by the user are candidates for serendipitous recommendations.

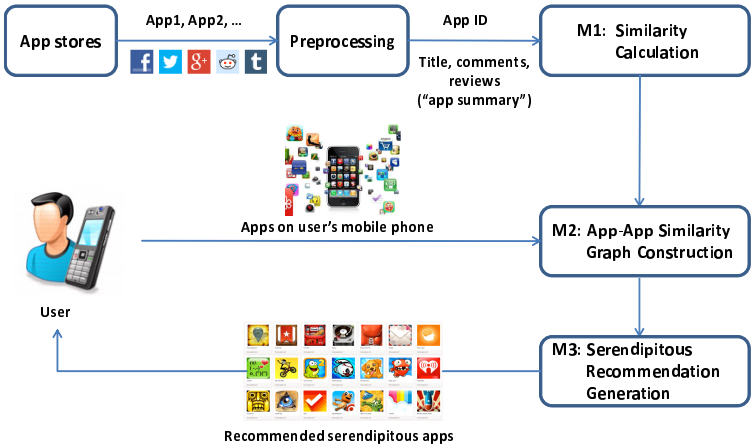

Fig. 1. System overview

3.2 Details of Our Proposed Method

Figure 1 shows an overview of our approach, consisting of the following three modules:

M1: Similarity calculation,
M2: App-app similarity graph construction, and
M3: Recommendation generation.

The main idea is that if two apps are connected by a path with highly weighted edges then these apps are also similar. Simply put, if there exists a path connecting two apps and the weights on each edge that constitutes this path are all sufficiently large, apps along this path which are not already downloaded by the user are good candidates for serendipitous recommendations to the user. The approach we have deployed is primarily for "casual discovery" of apps. We believe that, if a user specifically wants an app for a particular need, the user can download from the app store by searching for appropriate keywords. The only reason why a user would be interested in getting serendipitous recommendations is if the user is looking for interesting apps to try out. Thus, this approach is not aimed at finding apps that score high on the accuracy metric by being closer to user's interests but it scores high on the novelty and diversity from user's interest hence catering to serendipity. Apart from the modules mentioned above, we have preprocessing components for apps. These components are handled offline in order to reduce the computational cost of generating recommendations. At a high level, apps represent a set of apps from the app store. This subset of apps has been selected as apps that have been rated at least five times from our user database. Collecting actual usage data is the most ideal way to go about in generating recommendations. However, it is also difficult to collect real time usage data. Hence, we use comments or reviews as a substitute to narrow down on the subset of apps for this work (see "M1: Similarity Calculation" below for further details). Preprocessing, on the other hand, refers to a step that prepares data for similarity calculation. These can be truncations by removing punctuation, stop words, etc. The inputs from a target user are some apps that

have already been installed on the user's mobile phone. The apps are used after M2 has completed constructing app-app similarity graphs to generate recommendations in M3 for the target user. In the following, we detail the three modules. Note that modules M1 and M2 aim at discovering apps while M3 aims at generating recommendations.

M1: Similarity Calculation
While item-based CF method computes similarities between item pairs by using rating patterns given to these two items by different users, our approach computes similarities between two apps by using meta-information about the apps as discussed earlier. Thus, input to this module is metadata such as app ID, title of the app and comments or reviews of apps obtained after preprocessing. While we use app ID as an index to keep track of the apps, we use app title, app description and comments (hereafter, "app summary") to construct feature vectors of apps. Then, we calculate similarity between apps.

For each app a, we transform a into a feature vector $\boldsymbol{f}^a$ as follows:

$$\boldsymbol{f}^a = (w^a_{t_1}, w^a_{t_2}, \cdots, w^a_{t_m}), \tag{1}$$

where m is the number of distinct terms in app summary, and t_k $(k = 1, 2, \cdots, m)$ denotes each term. Using TF-IDF [19] scheme, we also define each element $w^a_{t_k}$ of $\boldsymbol{f}^a$ in Equation (1) as follows:

$$w^a_{t_k} = \frac{tf(t_k, a)}{\sum_{s=1}^{m} tf(t_s, a)} \cdot \log \frac{N_a}{df(t_k)},$$

where $tf(t_k, a)$ is the frequency of term t_k in the app summary, N_a is the total number of apps to recommend, and $df(t_k)$ is the number of app summary in which term t_k appears. Using feature vector for app defined by Equation (1), our approach computes similarity $sim(\boldsymbol{f}^{a_i}, \boldsymbol{f}^{a_j})$ between app a_i and a_j $(i \neq j)$ by Equation (2):

$$sim(\boldsymbol{f}^{a_i}, \boldsymbol{f}^{a_j}) = \frac{\boldsymbol{f}^{a_i} \cdot \boldsymbol{f}^{a_j}}{|\boldsymbol{f}^{a_i}||\boldsymbol{f}^{a_j}|}. \tag{2}$$

We consider all the apps in pair wise manner and generate similarity scores between them. The output is a list of similarity scores for each pair of apps ranging from 0 to 1. We employ cosine similarity since it is widely used and is effective in calculating similarity for short-text. For example, if we have 10 apps in our dataset, information about these 10 apps is first obtained from the app store, indexed and then cosine similarities are computed for ${}_{10}C_2$=45 app-pairs. These are then stored in a separate database to be used by the next module.

M2: App-App Similarity Graph Construction
The input to this module is a list of apps and the similarity scores between them computed by Equation (2) in M1. Let $G = (V, E)$ be an undirected and weighted graph, where V and E are a set of apps and a set of edges, respectively. Our approach creates an edge between two apps if the similarity between them defined by Equation (2) is greater than a predefined threshold. The similarity score is also used as a weight of the created edge. In order to control the size of the graph, we consider the top 30 most similar apps for every app. The output is app-app similarity graph that has vertices with the

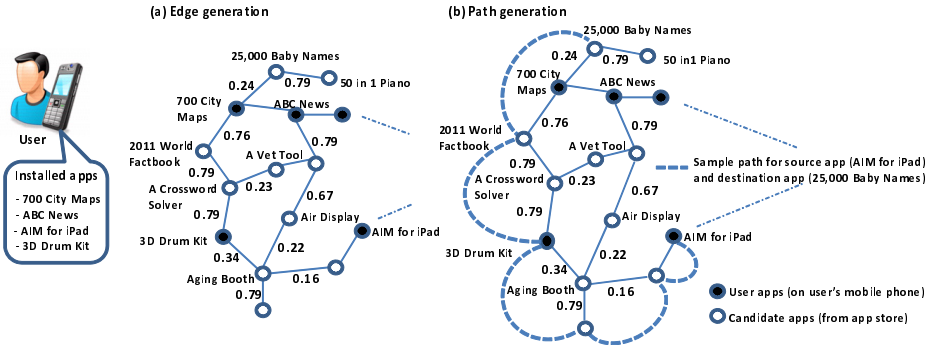

Fig. 2. Generation of (a) edge and (b) path from a given source (user's apps) to destination (candidate apps)

app IDs and edges with similarity scores between the vertices. Figure 2(a) shows the apps as vertices and edges (solid line) that connect the vertices marked with similarity between the apps.

M3: Serendipitous Recommendation Generation

At this stage, we have constructed app-app similarity graph that includes all the apps from our dataset. Now, we use the list of apps in a user's phone to start constructing paths from one app on the user's phone to another. In other words, we consider each app pair on a user's phone as source and destination for calculating paths between them. A path simply connects one app vertex to another which has similarity score above a threshold. We set the threshold to 0.4. Dotted line in Figure 2(b) illustrates the paths. By doing this, our approach can prevent each user's interests from too much drift, which commonly happens as the diversity of recommendations increases. It would not be useful to give highly diverse but less relevant apps to the user. Hence, to keep the interest of a user, we consider the apps installed on the user's phone to be the source and destination for the path construction. Adomavicius and Kwon [2] maximized the aggregate diversity of recommendations using max-flow algorithms in graph theory. Inspired by their approach, we find the application of the shortest-path finding algorithms to be extremely useful in the context of finding serendipitous items in the domain of mobile apps. The shortest-path algorithms aim to reduce the overall cost to traverse from a given source node to given destination node. In the app-app similarity graph, however, this cost is represented as the similarity between apps. Thus, during construction of the path, we take edges with low similarity to reach the destination, constructing the shortest path that connects the two apps. Paths are constructed for all the app-pairs on a user's phone. If an app-pair does not have a path connecting them, we eliminate such useless app-pair since it indicates lack of transitive relation between the two. The path construction step simply represents the app discovery problem and is useful for generating candidates for serendipitous recommendations. Thus, the output of this module is a list of serendipitous recommendations on the basis of apps installed on a target user's phone. This module thus generates recommendations by finding paths amongst the installed apps thus finding serendipitous apps to reduce over-specialization in recommendation lists.

4 Experiments

4.1 Experimental Data

Our dataset consists of 66,223 apps and 22,213 users in total, collected through a commercial project – Mobilewalla[3] [9]. Mobilewalla is a venture capital backed company that specializes in collecting, analyzing and presenting data related to mobile apps in four native stores, Apple iTunes[4], Google Android market[5], Blackberry native store[6] and Windows App store[7]. We implemented all modules with Java 1.7 and used MySQL v5.1 to store the similarity scores and recommendation paths. All modules and the database reside in the same computer (CPU: Intel CORE i5 2.27 GHz, Memory: 8 GBytes, OS: Windows 7).

4.2 Evaluation Measure

As described in Section 3.1, we define our serendipitous recommendation as the one that provides diverse and new apps. Thus, firstly, we evaluate our recommendations using normalized version of item novelty metric. Zhang and Hurley [26] introduced an item novelty measure in the course of their investigation on diversifying recommendation lists. Their approach, however, has a drawback; the measure can grow unbounded when the distance between two items becomes large. As such, we normalize the distance, $d(\boldsymbol{UMP}_{apps}, \boldsymbol{R}_{apps}^{srdp_j})$ between apps in user's mobile phone $\boldsymbol{UMP}_{apps}$ and serendipitous apps recommended to the user $\boldsymbol{R}_{apps}^{srdp_j}$ against the maximum distance of $d(\boldsymbol{UMP}_{apps}, \boldsymbol{R}_{apps}^{srdp_j})$. This normalized item novelty measure, which we denote as $nITN_{\boldsymbol{R}_{apps}^{srdp_j}}$, is thus defined as follows:

$$nITN_{\boldsymbol{R}_{apps}^{srdp_j}} = \frac{1}{N} \sum_{j=1}^{N} \frac{d(\boldsymbol{UMP}_{apps}, \boldsymbol{R}_{apps}^{srdp_j})}{\max d(\boldsymbol{UMP}_{apps}, \boldsymbol{R}_{apps}^{srdp_j})}, \tag{3}$$

where N is the number of the recommended apps (in which we have set N=5, 10, and 20) and we refer to them as $nITN@5$, $nITN@10$, $nITN@20$, respectively. If the recommendation list contains apps similar (dissimilar) to user apps, this measure has a smaller (larger) value. Larger results indicate more surprise, resulting in serendipitous recommendations. We also employ another metric, diversity-in-top-N, which is defined as follows:

$$\text{diversity-in-top-}N = \frac{\left|\bigcup_{u \in U} L_N(u)\right|}{|U|}, \tag{4}$$

where u, U, and $L_N(u)$ denote a user, the set of all users, and the list of N items recommended for user u, respectively. This metric measures the aggregate diversity

[3] http://www.mobilewalla.com
[4] http://store.apple.com/us
[5] https://play.google.com/store?hl=en
[6] http://appworld.blackberry.com/
[7] http://www.windowsphone.com/en-us/store

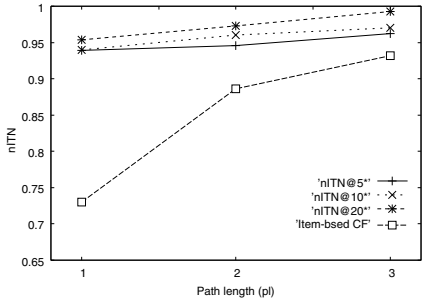

Fig. 3. Recommendation accuracy evaluated with $nITN@N$ (N=5, 10, 20) obtained by varying path length. ("*" denotes the difference between the results obtained by our approach and "item-based CF" is statistically significant for $p < 0.05$.)

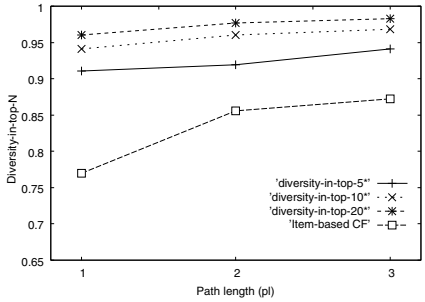

Fig. 4. Recommendation accuracy evaluated with diversity-in-top-N (N=5, 10, 20) obtained by varying path length. ("*" denotes the difference between the results obtained by our approach and "item-based CF" is statistically significant for $p < 0.05$.)

using the total number of distinct items amongst the top-N items recommended across all users [2]. This measure also has the drawback in that it can grow unbound when the number of target users becomes large. As such, we normalize the total number of distinct recommended items for all users against the set of all users.

4.3 Experimental Results

We employ item-based collaborative filtering [20] as our baseline system. Item-based collaborative filtering works by calculating similarity between items. This approach regards two items as similar if users give similar ratings to items. We set threshold of similarity to 0.4, and then select all items (apps) that are less than the threshold as neighbors to predict ratings of apps. Our goal is to recommend serendipitous recommendation. That is why we select low similarity neighboring apps. Since only a handful of apps receive ratings by users, this method recommends a small pool of apps over and over again, contributing to the "rich gets richer" phenomenon. We employ binary ratings for the baseline system, namely, 1 and 0 for installed and uninstalled app, respectively. Figure 3 shows normalized item novelty (nITN) of recommended serendipitous apps. Higher nITN score indicates larger serendipity. At path length, pl=3, our method outperforms the baseline (item-based collaborative filtering), and achieves the best with nITN of 0.993 for the top 20 apps for all users. To measure the diversity, we evaluated recommended apps with diversity-in-top-N [2]. Figure 4 shows the top N diversity (N= 5, 10, and 20). Similarly, at pl=3, our method outperforms the baseline and achieves the best for the top 20 apps for all users.

As we mentioned in Section 3.2, we employed cosine similarity in module M1. On the other hand, Pearson correlation can be one of alternatives as similarity measure. However, we observe that Pearson correlation gives identical results with cosine similarity. Furthermore, both nITN@5 in Figure 3 and diversity-in-top-5 in Figure 4 outperform our baseline system, item-based CF. A two-tailed t-test at the all path length shows that the difference between the results obtained by our approach and "item-based CF" is statistically significant for $p < 0.05$.

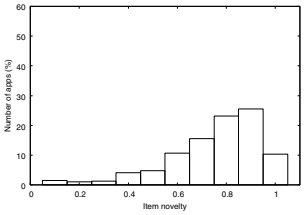

Fig. 5. Distribution of item novelty for User 1 (15 apps)

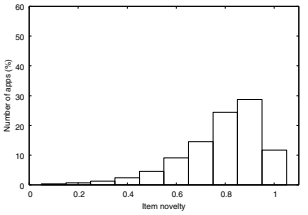

Fig. 6. Distribution of item novelty for User 2 (7 apps)

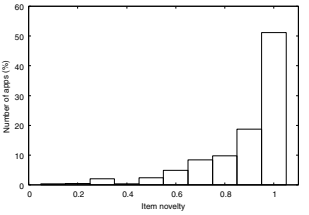

Fig. 7. Distribution of item novelty for User 3 (40 apps)

Having established the novelty and diversity of the recommended apps, we analyze the obtained results of recommendation qualitatively. A major challenge faced here was the lack of metrics to measure the new apps recommended. Since we do not have any prior data in terms of downloads or ratings for these apps, we cannot definitely say that which apps are relevant to each user, but by controlling the similarity threshold (we consider the top 30 most similar apps while graph construction), our approach can prevent user's interest from significant drift.

Figures 5, 6, and 7 show the distribution of item novelty for three sample users with various number of applications installed on their mobile phones when nITN@20 at pl=3. It ranges from 0.1 to 1.0, where 1.0 represents most diverse. Users 1 and 2 have installed 15 and 7 apps on each of their mobile phones, respectively. Our method can recommend about 30% and 25% diverse apps with $n\text{ITN} > 0.9$ to User 1 and User 2, respectively. User 3 has 40 apps installed on his phone and 50% of the recommendations generated have $n\text{ITN} > 0.9$. This indicates that, the higher the number of apps with the user, the greater the diversity of apps recommended, achieving a more desired novelty distribution. However, this does not mean that a smaller number of apps with the user would reduce the chances of being recommended diverse apps. Although Users 1 and 2 installed smaller number of apps compared with User 3, they can still obtain a notable novelty distribution score.

5 Conclusion

In this paper, we have developed a method for providing serendipitous recommendation for mobile apps by discovering highly diverse apps. Our approach captures user's preferences from apps already installed on the user's mobile phone and provides serendipitous recommendation by constructing app-app similarity graph. Our evaluation is still not complete as we need to develop additional metrics for verifying the accuracy of these serendipitous recommendations. This is a challenging task since any metrics for serendipitous recommendation have not been developed so far. While we focused on the domain of mobile apps, our approach can be applied to any domains with huge cardinality. In future work, we plan to justify the recommendations with newly developed metrics that can evaluate serendipitous recommendations.

References

1. Adamopoulos, P., Tuzhilin, A.: On Unexpectedness in Recommender Systems: Or How to Better Expect the Unexpected. Technical Report CBA-13-03, Stern School of Business, New York University (2013)
2. Adomavicius, G., Kwon, Y.: Maximizing Aggregate Recommendation Diversity: A Graph-Theoretic Approach. In: Proc. of the 1st International Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011), pp. 3–10 (2011)
3. Aggarwal, C.C., Wolf, J.L., Wu, K.-L., Yu, P.S.: Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In: Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999), pp. 201–212 (1999)
4. Andre, P., Schraefel, M.C., Teevan, J., Dumais, S.T.: Discovery is Never by Chance: Designing for (Un)Serendipity. In: Proc. of the 7th SIGCHI Conference on Creativity and Cognition (C&C 2009), pp. 305–314 (2009)
5. Andre, P., Teevan, J., Dumais, S.T.: From X-Rays to Silly Putty via Uranus: Serendipity and its Role in Web Search. In: Proc. of the 27th International Conference on Human Factors in Computing Systems (CHI 2009), pp. 2033–2036 (2009)
6. Basu, C., Hirsh, H., Cohen, W.: Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In: Proc. of the 15th National Conference on Artificial Intelligence (AAAI 1998), pp. 714–720 (1998)
7. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Proc. of the 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998), pp. 43–52 (1998)
8. Costa-Montenegro, E., Barragáns-Martínez, A.B., Rey-López, M.: Which App? A Recommender System of Applications in Markets: Implementation of the Service for Monitoring Users' Interaction. Expert Systems with Applications: An International Journal 39(10), 9367–9375 (2012)
9. Datta, A., Dutta, K., Kajanan, S., Pervin, N.: Mobilewalla: A Mobile Application Search Engine. Mobile Computing, Applications, and Services 95(5), 172–187 (2012)
10. Davidsson, C., Moritz, S.: Utilizing Implicit Feedback and Context to Recommend Mobile Applications from First Use. In: Proc. of the 2011 Workshop on Context-awareness in Retrieval and Recommendation (CaRR 2011), pp. 19–22 (2011)
11. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.B.: Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM 35(12), 61–70 (1992)
12. Kawamae, N.: Serendipitous Recommendations via Innovators. In: Proc. of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010), pp. 218–225 (2010)
13. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM 40(3), 77–87 (1997)
14. Lathia, N., Hailes, S., Capra, L., Amatriain, X.: Temporal Diversity in Recommender Systems. In: Proc. of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010), pp. 210–217 (2010)
15. Lin, J., Sugiyama, K., Kan, M.-Y., Chua, T.-S.: Addressing Cold-Start in App Recommendation: Latent User Models Constructed from Twitter Followers. In: Proc. of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013), pp. 283–292 (2013)

16. Nakatsuji, M., Fujiwara, Y., Tanaka, A., Uchiyama, T., Fujimura, K., Ishida, T.: Classical Music for Rock Fans?: Novel Recommendations for Expanding User Interests. In: Proc. of the 19th International Conference on Information and Knowledge Management (CIKM 2010), pp. 949–958 (2010)
17. Resnick, P., Iacovou, N., Suchak, M., Bergstorm, J.R.P.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proc. of the ACM 1994 Conference on Computer Supported Cooperative Work (CSCW 1994), pp. 175–186 (1994)
18. Ricci, F., Shapira, L., Kantor, B.: Recommender Systems Handbook. Springer (2011)
19. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill (1983)
20. Sarwar, B.M., Karypis, G., Konstan, J., Riedl, J.: Item-Based Collaborative Filtering Recommendation Algorithms. In: Proc. of the 10th International World Wide Web Conference (WWW10), pp. 285–295 (2001)
21. Sarwar, B.M., Karypis, G., Konstan, J.A.: Analysis of Recommendation Algorithms for E-commerce. In: Proc. of the 2nd ACM Conference on Electronic Commerce (EC 2000), pp. 158–167 (2000)
22. Sugiyama, K., Kan, M.-Y.: Serendipitous Recommendation for Scholarly Papers Considering Relations Among Researchers. In: Proc. of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries (JCDL 2011), pp. 307–310 (2011)
23. Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J., Venkataraman, S.: Identifying Diverse Usage Behaviors of Smartphone Apps. In: Proc. of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC 2011), pp. 329–344 (2011)
24. Yan, B., Chen, G.: AppJoy: Personalized Mobile Application Discovery. In: Proc. of the 9th International Conference on Mobile Systems, Applications and Services (MobiSys 2011), pp. 113–126 (2011)
25. Yin, P., Luo, P., Lee, W.-C., Wang, M.: App Recommendation: A Contest between Satisfaction and Temptation. In: Proc. of the 6th International Conference on Web Search and Data Mining (WSDM 2013), pp. 395–404 (2013)
26. Zhang, M., Hurley, N.: Avoiding Monotony: Improving the Diversity of Recommendations. In: Proc. of the 2008 ACM Conference on Recommender Systems (RecSys 2008), pp. 123–130.
27. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving Recommendation Lists Through Topic Diversification. In: Proc. of the 14th International World Wide Web Conference (WWW 2005), pp. 22–32 (2005)