Mining Usage Traces of Mobile Apps for Dynamic Preference Prediction

Zhung-Xun Liao[1], Wen-Chih Peng[1], and Philip S. Yu[2]

[1] Department of Computer Science and Information Engineering, National Chiao Tung University, HsinChu, Taiwan, ROC
[2] Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

Abstract. Due to a huge amount of mobile applications (abbreviated as Apps), for Apps providers, the usage preferences of Apps are important in recommending Apps, downloading Apps and promoting Apps. We predict and quantize users' dynamic preferences by exploring their usage traces of Apps. To address the dynamic preference prediction problem, we propose Mode-based Prediction (abbreviated as MBP) and Reference-based Prediction (abbreviated as RBP) algorithms. Both MBP and RBP consist of two phases: the trend detection phase and the change estimation phase. In the trend detection phase, both algorithms determine whether the preference of an App is increasing or decreasing. Then, in the change estimation phase, the amount of preference change is calculated. In particular, MBP adopts users' current usage mode (active or inactive), and then estimates the amount of change via our proposed utility model. On the other hand, RBP calculates an expected number of usage as a reference, and then builds a probabilistic model to estimate the change of preference by comparing the real usage and the reference. We conduct comprehensive experiments using two App usage traces and one music listening log, the Last.fm dataset, to validate our proposed algorithms. The experimental results show that both MBP and RBP outperform the usage-based method that is based solely on the number of usages.

Keywords: Dynamic Preference Prediction, Mobile Applications, Apps.

1 Introduction

As mobile devices become more and more popular, a tremendous amount of mobile applications (abbreviated as Apps) are designed for varied functions and purposes. Users can download and execute Apps in their mobile devices to satisfy their needs and affinities. For App providers, to understand users' preferences is quite important to recommend new Apps, and to decide their marketing strategies for selling Apps [1–3]. Although users can rate the Apps they have experienced, only a small percentage of users rate their Apps. For example, the famous App, Angry Birds, only received 4% ratings from 1.3 million of downloads [4]. Besides, users may not be willing to consistently rate the Apps when they change their preferences. On the other hand, although [5] states an Apps

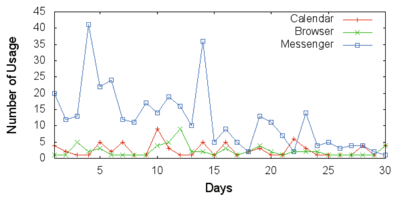


Fig. 1. The number of usages of different mobile applications

recommendation problem, it does not show the dynamic preferences of users. By contrast, through the dynamic preferences, we can not only recommend Apps but investigate more tasks on Apps.

In this paper, we aim to predict users' dynamic preferences of each App and further quantize the preferences to real numbers such that we can compare the preferences among different users. As users repeatedly invoke these Apps, their preferences are dynamic over time based on what they have experienced. Here, we claim that a user's dynamic preference is related to the usage trace (i.e., series of usage counts). For example, Fig. 1 shows three usage traces of Calender, Browser, and Messenger, for a certain user. As can be seen in Fig. 1, the number of usages on "Messenger" apparently drops down after 14 days (two weeks). Therefore, we can infer that the user decline his/her preference on "Messenger" in 14 days by either implicit or explicit reasons.

Nevertheless, usage counts of Apps are not directly related to the preferences of Apps. For example, in Fig. 1, although the usage count of Messenger is higher than the other two Apps, the preference of Messenger is not necessarily higher than the other two Apps. Probably, Messenger is a communication tool, which is designed to be used frequently. As for Calendar, users will not frequently check their calendar all the time. In our experimental results, we also show that the usage-based algorithm cannot predict anything, where its accuracy is often close to zero. To correctly predict preferences of Apps from the usage traces, we propose two methods, Mode-based Prediction (abbreviated as MBP) and Reference-based Prediction (abbreviated as RBP). Both methods utilize different strategies to avoid the impact of the inherent magnitude bias of the the usage counts. MBP adopts only the usage mode of Apps: active mode for using the App while inactive mode for not using the App. RBP refers to the previous usage counts of each App as a reference history and thus, the usage count becomes a relative value of the reference history.

Both MBP and RBP consist of two phases: the trend detection phase and the change estimation phase. The first phase determines whether the preference is decreasing or increasing. The second phase estimates the absolute value on the preference change. For MBP, we increase the preferences of those Apps which are used at current time unit, but decreases the preferences of others. Then, we propose a utility model: when a user uses more Apps at the same time unit, each App would receive less preference increment. According to the utility model, we can calculate the increment and decrement of each App. For RBP, it calculates an expected number of usage for each App at current time unit by solving an optimization problem where the expected number of usage can keep the trend of preference change staying static. If the actual number of usage is larger (smaller, respectively) than the expectation, the preference will increase (decrease, respectively). Then, RBP uses a probabilistic model to estimate the change of preferences.

The contributions of this study are:

1. We explore usage traces of Apps for dynamically predicting the perferences of Apps.
2. We analyze the characteristics of Apps, and propose two algorithms, MBP and RBP, to predict preferences of Apps.
3. In the MBP method, we derive the dynamic preferences according to only the usage mode and propose a utility model to calculate the change of users' preferences.
4. In the RBP method, by solving an optimization problem, the expected number of usage is derived as a reference, and a probabilistic model is constructed to estimate the users' preferences.
5. We conduct a comprehensive performance evaluation. The experimental results show that the predicted dynamic preferences of both MBP and RBP can better reflect users' behavior

2 Related Work

To the best of our knowledge, this paper is the first work discussing dynamic preferences prediction problem. Although there are many research works discussing the problem of predicting users' preference, they only focused on a static environment. In a static circumstance, such as renting movies and purchasing books, users generally only act on them once and the preference remain static. Therefore, they can use the existing user preferences to predict the unknown preferences through the attributes of items [6]. The attributes could be the metadata, such as artist, genre, etc., or the ratings the item already had. Although [6] focused on predicting the ratings of musics, they still treated the music ratings as static preferences. This is because their focus is on purchasing songs or CDs, not on the preference to listening to a song from a user collection at a particular moment. Only the authors in [7–9] recognized the temporal dynamics of users' preferences. Nevertheless, [7] still need to obtain at least a portion of static ratings as training data. [8, 9] only consider the evolution of users' behavior, instead of quantize their preferences. For predicting preferences of Apps, users can use Apps repeatedly; therefore, their preference changes over time, and even be impacted by new Apps [10]. Consequently, the traditional preference prediction methods cannot be adopted for the dynamic preference problem, because 1) the traditional methods all need to obtain at least a portion of static preferences as training data, and 2) the static preferences are out-of-date when we perform the prediction in a dynamic environment.

3 Preliminary

In this section, we first describe the symbols used in this paper. Explicitly, we use r_{min} and r_{max} to represent the minimum and maximum value of users' preference. Thus, the preference at time unit t is a real number $r_{ij}^{(t)} \in [r_{min}, r_{max}]$, which represents the preference of user i on App j. To facilitate the presentation of this paper, U is the set of users and I is the set of Apps. A dynamic preference matrix is used to represent the preferences of Apps at a certain time unit. Here, we divide time space into time units, and use l, an application dependent parameter, to represent the length of a time unit. The formal definition is below:

Definition 1. ***(Dynamic Preference Matrix)*** *A dynamic preference matrix at time unit t, $R^{(t)}$, is a $|U| \times |I|$ matrix, where $r_{ij}^{(t)} \in [r_{min}, r_{max}]$, for each $r_{ij}^{(t)}$.*

A usage count matrix constructed from users' traces is defined in Definition 2

Definition 2. ***(Usage Count Matrix)*** *A usage count matrix at time unit t, $C^{(t)}$, is a $|U| \times |I|$ matrix, where each element $c_{ij}^{(t)}$ represents how many times user i used App j at time unit t.*

We use a change matrix to record the preference change of each user-App pair. When the value is positive (negative, respectively), the preference is increasing (decreasing, respectively). Definition 3 shows the detail of change matrix. In this paper, the change matrix is derived from both usage count matrix and dynamic preference matrix.

Definition 3. ***(Change Matrix)*** *A change matrix at time unit t, denoted as $\Delta^{(t)}$, is a $|U| \times |I|$ matrix, where the value of each element $\delta_{ij}^{(t)}$ is in either $[0, r_{max} - r_{ij}^{(t-1)}]$ for positive value, or in $[r_{ij}^{(t-1)} - r_{min}, 0]$ for negative value.*

We claim that the preference of an App would not change dramatically. Even when users do not use an App for a long time, the preference of it would decay smoothly over time. Therefore, we derive users' preferences according to the previous preferences and current usage behavior as described in Definition 4.

Definition 4. ***(Dynamic Preferences Prediction Problem)*** *Let $R^{(t-1)}$ be the dynamic preference matrix at time unit $t-1$, and $C^{(t)}$ be the usage count matrix at time unit t, the dynamic preference prediction problem is 1) calculating the change matrix, $\Delta^{(t)}$, and 2) deriving $R^{(t)}$ according to Eq. 1.*

$$R^{(t)} = R^{(t-1)} + \Delta^{(t)} \tag{1}$$

For example, suppose we have two users and three Apps, and the system parameters are $r_{min} = 0$ and $r_{max} = 5$. Let $R^{(t-1)} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}$ be the dynamic preference matrix derived at time unit $t-1$, and $C^{(t)} = \begin{bmatrix} 100 & 2 & 30 \\ 2 & 300 & 40 \end{bmatrix}$ be the usage count matrix. First, we calculate the change matrix according to $C^{(t)}$ and $R^{(t-1)}$, such that, in this example, the values of the change matrix are related to the usage counts and will be in the defined range to avoid the values in $R^{(t)}$ being out of range. Assume that we can obtain $\varDelta^{(t)} = \begin{bmatrix} 4 & -1 & 1 \\ -2 & 2 & 1 \end{bmatrix}$. Then, the new dynamic preference could be derived as $R^{(t)} = \begin{bmatrix} 5 & 1 & 4 \\ 0 & 5 & 5 \end{bmatrix}$.

4 Dynamic Preference Prediction

As described in Definition 4, to obtain the dynamic preference matrix, $R^{(t)}$, we need to know the change matrix, $\varDelta^{(t)}$, in advance. Here, we use δ_{ij}^{t} to represent the elements in $\varDelta^{(t)}$. Empirically, we can calculate δ_{ij}^{t} by Eq. 2 which consists of two parts: 1) $m \in \{0, 1\}$ which indicates whether δ_{ij}^{t} is positive ($m = 0$) or negative ($m = 1$), and 2) $v_{ij}^{(t)} > 0$ which is the absolute value of δ_{ij}^{t}. Through this equation, we can calculate the change matrix, $\varDelta^{(t)}$, by finding a proper pair of m and v_{ij}^{t} for each $\delta_{ij}^{(t)}$.

$$\delta_{ij}^{(t)} = (-1)^{m} \times v_{ij}^{(t)} \tag{2}$$

In this paper, we design a two-phase framework: the trend detection phase for the value of m and the change estimation phase to calculate $v_{ij}^{(t)}$. In order to smooth the preference change, the value of $v_{ij}^{(t)}$ depends on not only the current usage count, $c_{ij}^{(t)}$ but the previous preference, $r_{ij}^{(t-1)}$. In addition, when the preference is increasing (respectively, decreasing), the value of $v_{ij}^{(t)}$ is in the range of [0, $r_{max} - r_{ij}^{(t-1)}$] (respectively, [0,$r_{ij}^{(t-1)} - r_{min}$]). Thus, we can formulate $v_{ij}^{(t)}$ as in Eq. 3, where $u_{ij}^{(t)}$ is a utility parameter determined by user's preference change. Explicitly, when $u_{ij}^{(t)}$ is larger (i.e. user's preference change is large), $v_{ij}^{(t)}$ would be larger.

In order to address the challenge related to the number of usages of Apps, we propose two algorithms based on different points of view. The first one is Mode-based Prediction (MBP) which takes into account of the binary usage mode of active and inactive. The second one is called Reference-based Prediction (RBP) which adopts the previous usage counts as a reference history to examine the $\varDelta^{(t)}$ matrix.

$$v_{ij}^{(t)} = \begin{cases} (r_{ij}^{(t-1)} - r_{min}) \times u_{ij}^{(t)} & ,m = 1 \\ (r_{max} - r_{ij}^{(t-1)}) \times u_{ij}^{(t)} & ,m = 0 \end{cases} \tag{3}$$

4.1 Mode-Based Prediction (MBP)

The Mode-based Prediction (MBP) ignores the magnitude of usage counts by only considering two usage mode: one is active mode for using the App and another is inactive mode for not using the App. Then, a utility model is proposed to measure the usage change of a user, and the $\Delta^{(t)}$ matrix could be estimated through this model.

The trend detection phase. In this phase, we decide the value of m in Eq. 2. If user i executed App j at time unit t (i.e. $c_{ij}^{(t)} > 0$), we would set m as 0 (increase the preference). By contrast, if $c_{ij}^{(t)} = 0$, we set m to 1 (decrease the preference).

$$\sum_{k \in P} u_{ik}^{(t)} - \sum_{k \in N} u_{ik}^{(t)} = 0 \tag{4}$$

$$u_{ij}^{(t)} = \begin{cases} \frac{1}{|P|} & , c_{ij}^{(t)} > 0 \\ \frac{1}{|N|} & , c_{ij}^{(t)} = 0 \end{cases} \tag{5}$$

$$v_{ij}^{(t)} = \begin{cases} \frac{r_{max} - r_{ij}^{(t-1)}}{|P|} & , c_{ij}^{(t)} > 0 \\ \frac{r_{ij}^{(t-1)} - r_{min}}{|N|} & , c_{ij}^{(t)} = 0 \end{cases} \tag{6}$$

The change estimation phase. The second phase is to estimate the absolute value of the preference change. In other words, we need to derive the value of $v_{ij}^{(t)}$ according to utility parameter, $u_{ij}^{(t)}$. Since we only have the information of usage mode of each App, we propose a utility model to derive the utility parameter based only on the usage mode. Intuitively, when a user spends more time on some Apps, (s)he should spend less time on others. Thus, we claim that the overall usage change among Apps should be equal to 0. Eq. 4 formulates the utility model, where P (respectively, N) is the set of Apps with active (respectively, inactive) mode. Suppose that the importance of each App is the same, the utility parameter is derived by Eq. 5. As a result, we can obtain $\delta_{ij}^{(t)}$ from $u_{ij}^{(t)}$, as shown in Eq. 6.

4.2 Reference-Based Prediction (RBP)

Although MBP successfully avoids the magnitude of usage counts by adopting the usage mode and the utility model, ignoring the magnitude of the usage counts makes the estimated preferences not be able to reflect users' actual preferences. For example, in Fig. 2, the preference of Messenger predicted by MBP becomes higher and higher over time, since MBP increases the user's preference once the user invokes the Apps. However, we believe that the curve representing the preference of Messenger should be like the Ideal one. To obtain the ideal result, we propose a Reference-based Prediction (RBP) algorithm which compares the usage counts within an App instead of with other Apps.

RBP uses the previous usage counts of each App as a reference history, and derives a reference value from the reference history. In this paper, the size of reference history is decided by a tunable parameter, h, which means how many

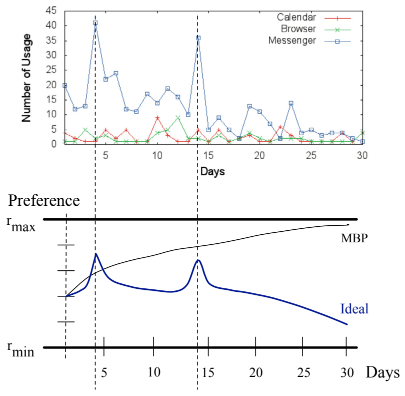


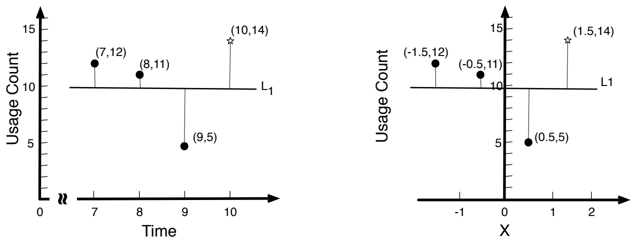
Fig. 2. The preferences derived by MBP comparing with the Ideal preferences

historical data points are included into the reference history. The concept is that only when the actual usage count of an App is higher than the reference value, its preference is increasing. Similarly, the preference decreases only when the number of usage is less than the reference value.

$$slope = \frac{(h+1)\sum_{k=t-h}^{t} k \times c_{ij}^{(k)} - \sum_k k \sum_k c_{ij}^{(k)}}{(h+1)\sum_k k^2 - (\sum_k k)^2} = 0 \tag{7}$$

$$(h+1)\sum_{k=t-h}^{t} k \times c_{ij}^{(k)} - \sum_k k \sum_k c_{ij}^{(k)} = 0 \tag{8}$$

The trend detection phase. In this phase, we decide whether the value of $\delta_{ij}^{(t)}$ is negative or positive. Here, we use the previous usage counts as the reference history and derive an expected number of usage as the reference value from the reference history. We adopt the linear regression to model the trend of reference history, and thus, the expected number of usage count should make the slope of the regression line be zero. Since the slope of a regression line represents the trend of the data points, the expected number of usage count which makes the regression line stay horizontal means that it makes the preference stay static. Then, if the actual number of usage is larger (smaller, respectively) than the expected number of usage, the preference is considered as increasing (decreasing, respectively). We use Fig. 3(a) to illustrate the concept of obtaining the expected number of usage by linear regression model. In Fig. 3(a), the three black points are the reference history (i.e. $h = 3$) and the reference value is the expected usage at time unit 10 (marked as a star point) which makes the regression line, L_1, be horizontal. Therefore, the goal of this phase is to find the value of star point by satisfying Eq. 7 which can be simplified into Eq. 8.



(a) Deriving the star point from the three black points. (b) Shifting the data points in Fig. 3(a).

Fig. 3. Estimate the expected usage count (marked as a star point)

Since we only consider the slope of the regression line, we can shift the regression line left such that $\sum_{k=1}^{h+1} x_k = 0$, where x_k represents the shifted position in x-axis of the k-th point of reference history and thus, x_{h+1} is the shifted x-axis of the star point. As shown in Fig. 3(b), we can shift the regression line to the positions of x-axis as $< -1.5, -0.5, 0.5, 1.5 >$. Eq. 9 shows how to calculate the shifted x-axis positions. Now, we can simplify Eq. 8 into Eq. 10, where the index of time units of $c_{ij}^{(k)}$ is also shifted to $(k + t - h - 1)$ for $k = 1, 2, \ldots, h + 1$.

$$x_k = \frac{2k - (h + 2)}{2} \tag{9}$$

$$(h + 1) \sum_{k=1}^{h+1} x_k \times c_{ij}^{(k+t-h-1)} = 0 \tag{10}$$

Therefore, we can extract $c_{ij}^{(t)}$ from Eq. 10, and it is the expected number of usage $EXP(c_{ij}^{(t)})$, which could be derived from Eq. 11. For example, the value of the star point in Fig. 3(a) is $EXP(c_{ij}^{(10)}) = [(3 + 2)(12 + 11 + 5) - 2(1 \times 12 + 2 \times 11 + 3 \times 5)]/3 = 42/3 = 14$.

$$\begin{aligned} EXP(c_{ij}^{(t)}) &= -\frac{\sum_{k=1}^{h} x_k \times c_{ij}^{(k+t-h-1)}}{x_{h+1}} \\ &= -\frac{\frac{2(1)-(h+2)}{2} c_{ij}^{(t-h)} + \ldots + \frac{2(h)-h-2}{2} c_{ij}^{(t-1)}}{\frac{2(h+1)-(h+2)}{2}} \\ &= \frac{(h + 2) \sum_{k=1}^{h} c_{ij}^{(k+t-h-1)} - 2 \sum_{k} k \times c_{ij}^{(k+t-h-1)}}{h} \end{aligned} \tag{11}$$

The change estimation phase. As we have $EXP(c_{ij}^{(t)})$ to be the reference value, we need to formulate the utility parameter, $u_{ij}^{(t)}$, by calculating the distance between $c_{ij}^{(t)}$ and $EXP(c_{ij}^{(t)})$, denoted as $dist(EXP(c_{ij}^{(t)}), c_{ij}^{(t)})$. When $c_{ij}^{(t)}$ is far from $EXP(c_{ij}^{(t)})$, it means that the user is considered more likely to change his/her preference. Since we need a distance measure between 0 and 1, directly subtracting $EXP(c_{ij}^{(t)})$ from $c_{ij}^{(t)}$ or the other way around will not work. We devise the following distance measure. Here, $dist(EXP(c_{ij}^{(t)}), c_{ij}^{(t)})$ is estimated by evaluating how many possible cases are between $EXP(c_{ij}^{t})$ and $c_{ij}^{(t)}$. Therefore, when the preference is increasing ($m = 0$), we use $p(EXP(c_{ij}^{(t)}) \leq x \leq c_{ij}^{t})$ representing the probability of obtaining a number of usage in-between $EXP(c_{ij}^{(t)})$ and $c_{ij}^{(t)}$, where x is a random variable. On the other hand, when the preference is decreasing ($m = 1$), the distance between $EXP(c_{ij}^{(t)})$ and $c_{ij}^{(t)}$ is formulated as $p(c_{ij}^{(t)} \leq x \leq EXP(c_{ij}^{(t)}))$. In this paper, we approximate the probability, $p(c_{ij}^{(t)})$, of using an App j by $c_{ij}^{(t)}$ times in a given time duration l (a parameter for the length of each time unit) by assuming a Poisson distribution shown in Eq. 12, where $\lambda = EXP(c_{ij}^{(t)})$. Now, the utility parameter, $u_{ij}^{(t)}$, could be formulated as in Eq. 13 and the absolute amount of preference change, $v_{ij}^{(t)}$, as in Eq. 14. We also list algorithm 1 to describe the flow of RBP in detail. In the first iteration, we set $r_{ij}^{(t)}$ to an initial preference, r_{init}, which is a tunable parameter. The preference will stay the same when the actual number of usage equals to the expected number of usage.

$$p(c_{ij}^{(t)}) = \frac{\lambda^{c_{ij}^{(t)}} \times e^{-\lambda}}{c_{ij}^{(t)}!} \tag{12}$$

$$u_{ij}^{(t)} = dist(\lambda, c_{ij}^{(t)}) = \begin{cases} p(\lambda \leq x \leq c_{ij}^{(t)}) = \sum_{k=\lambda}^{c_{ij}^{(t)}} p(k) \ , c_{ij}^{(t)} > \lambda \\ p(c_{ij}^{(t)} \leq x \leq \lambda) = \sum_{k=c_{ij}^{(t)}}^{\lambda} p(k) \ , c_{ij}^{(t)} < \lambda \end{cases} \tag{13}$$

$$v_{ij}^{t} = \begin{cases} (r_{max} - r_{ij}^{(t-1)}) \times \sum_{k=\lambda}^{c_{ij}^{(t)}} p(k) \ , c_{ij}^{(t)} > \lambda \\ (r_{ij}^{(t-1)} - r_{min}) \times \sum_{k=c_{ij}^{(t)}}^{\lambda} p(k) \ , c_{ij}^{(t)} < \lambda \end{cases} \tag{14}$$

5 Experimental Results

To evaluate the accuracy of the derived dynamic preferences, we examine the accuracy by testing the performance of using those derived preferences to make

Algorithm 1. Algorithm of Reference-based Prediction

```
Input: Input: $R^{(t-1)}$, $C^{(t)}$
Output: Output: $R^{(t)}$
1  foreach $c_{ij}^{(t)}$ do
2      Let $r_{ij}^{(t)} \leftarrow r_{init}$
3  end
4  foreach initialled $r_{ij}^{(t-1)}$ do
5      $EXP(c_{ij}^{(t)}) \leftarrow \frac{(h+2)\sum_{k=1}^{h} C - 2\sum_{k} k \times c^{(k)}}{h}$
6      if $c_{ij}^{(t)} > EXP(c_{ij}^{(t)})$ then
7          $m \leftarrow 0$ $P \leftarrow P \cup App_j$
8      else
9          $m \leftarrow 1$ $N \leftarrow N \cup App_j$
10     end
11     $\lambda \leftarrow EXP(c_{ij}^{(t)})$
12     if $App_j \in P$ then
13         $v_{ij}^{(t)} \leftarrow (r_{max} - r_{ij}^{(t-1)}) \times \sum_{k=c_{ij}^{(t)}}^{\lambda} \frac{\lambda^{c_{ij}^{(t)}} \times e^{-\lambda}}{c_{ij}^{(t)}!}$
14     else
15         $v_{ij}^{(t)} \leftarrow (r_{ij}^{(t-1)} - r_{min}) \times \sum_{k=\lambda}^{c_{ij}^{(t)}} \frac{\lambda^{c_{ij}^{(t)}} \times e^{-\lambda}}{c_{ij}^{(t)}!}$
16     end
17     $\delta_{ij}^{(t)} \leftarrow (-1)^m \times v_{ij}^{(t)}$
18 end
19 return $R^{(t)} \leftarrow R^{(t-1)} + \Delta^{(t)}$
```

recommendation. We adopt the All-But-One evaluation methods [11] which, for each user, we iteratively skip one App from a user's preference list, and then make recommendation for this user. If the skipped App is recommended, we treat it as a hit. The hit ratio of user u at time unit t is calculated by Eq. 15, where k is the number of recommended Apps, $I(\cdot)$ is an indicator function defined in Eq. 16, $App_k(u,t)$ is the top-k Apps with highest preference score for user u at time unit t, and $R_k(u,t)$ is the list of k recommended Apps for user u at time unit t. The length of one time unit, l, is 1 day for both App traces, and 7 days for the Last.fm dataset. Eventually, the overall accuracy is the average hit ratio of every user at every time unit, which is shown in Eq. 17.

$$HitRatio_k(u,t) = \frac{\sum_{i \in App_k(u,t)} I_{R_k(u,t)}(i)}{|App_k(u,t)|} \tag{15}$$

$$I_{R_k(u,t)}(i) \begin{cases} 1 \ , i \in R_k(u,t) \\ 0 \ , i \notin R_k(u,t) \end{cases} \tag{16}$$

$$Accuracy_k = OverallHitRatio_k = \frac{\sum_u \sum_t HitRatio_k(u,t)}{|U| \times |T|} \tag{17}$$

5.1 Environment

The range of users' preferences is set to [0,5]. The adopted recommendation algorithm is Collaborative Filtering (CF) provided by Apache project, Mahout, with its similarity function as Pearson correlation function.

Dataset description. We have three real-world datasets: two are App usage traces and one is music listening log from Last.fm [12]. For the two traces of App usage, one is a smaller trace which consists of 30 users and 226 Apps, while the other one has 80 users and 650 Apps. Through the two different scales of datasets, we can ensure whether our methods are scalable or not. For the music listening dataset, we have a relatively huge amount of users in the Last.fm 1K-users dataset. The music listening dataset consists of 1000 users and 48,361 music albums which is a very sparse dataset we have to deal with. The total time duration for two App traces is half a year and for the music listening log is one and half years.

$$\frac{r_{ij}^t - r_{min}}{r_{max} - r_{min}} = \frac{c_{ij}^t - \min_k c_{ik}^t}{\max_k c_{ik}^t - \min_k c_{ik}^t} \tag{18}$$

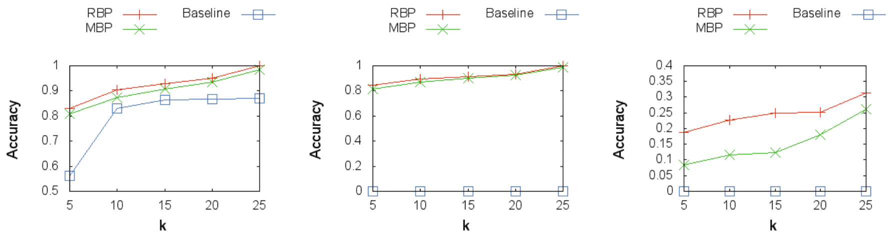
$$r_{ij}^t = \frac{(c_{ij}^t - \min_k c_{ik}^t) \times (r_{max} - r_{min})}{\max_k c_{ik}^t - \min_k c_{ik}^t} + r_{min} \tag{19}$$

Compared methods. To compare the accuracy of our proposed algorithms, we adopt a usage-based method as the baseline. The usage-based method calculates the users' preferences only by the usage count. The item with largest number of usage will be assigned the preference of r_{max}, while the one with smallest number of usage will be assigned the preference of r_{min}. Besides, the preferences of other items are calculated by an interpolation method shown in Eq. 18.

5.2 Performance Evaluation

In this study, we evaluate the accuracy under various number of recommended Apps, k, and different length of a time unit, l over two proposed algorithms and one baseline method. Then, we focus on the proposed Reference-based Prediction (RBP) algorithm to see the accuracy when changing the parameter h which is used to control how many historical data are used.

Accuracy changed by k. Since k would affect the hit ratio, we calculate the hit ratio by different k from 5 to 25. However, although larger k could derive a better performance on hit ratio, fewer recommended items is more meaningful for users. Figs. 4(a) and 4(b) show the results of two App traces under different numbers of recommended items, k. Obviously, the accuracy increases as k grows up. Specifically, when $k = 5$, both RBP and MBP can achieve the accuracy of more than 80%. we note that in Fig. 4(b), the baseline remains close to zero



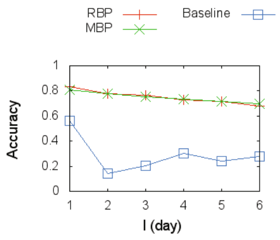
(a) App-small dataset. (b) App-large dataset. (c) Last.fm dataset.

Fig. 4. Accuracy evaluation with different k

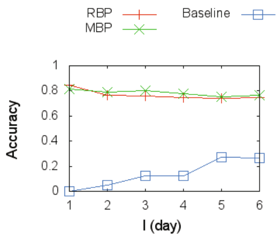
even for $k = 25$, while in Fig. 4(a), the baseline achieves relatively low accuracy compared with RBP for $k = 5$. This is because the App-large dataset consists of more Apps and makes the dataset become sparser than App-small. Fig 4(c) depicts the results of Last.fm dataset. Since the music listening dataset is much sparser than the App traces, the performance on accuracy is not as good as the accuracy of the App traces. However, RBP is always the best method, while the baseline is close to zero. Here, for the two App traces, the length of one time unit is one day and the size of reference history, h, of RBP is set to 4 time units; for music listening dataset, the length of time unit is 1 week and the parameter h of RBP is 6 time units.

Impact of parameter *l*. Here, we evaluate the accuracy change of various length of a time unit. As can be seen in Figs. 5(a) and 5(b), both RBP and MBP slightly decrease their accuracy when the amount of training data increases. The best length of a time unit is one day which matches the human behavior. By contrast, the baseline method increases the accuracy when the number of training data becomes larger. Because the baseline method does not consider the temporal information, more training data could provide more information to overcome this drawback. However, when $d > 6$, the accuracy of baseline method also declines. On the other hand, as shown in Fig. 5(c), the best length of a time unit for Last.fm dataset is 7 days (one week) since music listening behavior is sparse and users may repeat the songs they listened in one week. Here, the reference history parameter, h, is set to 6 time units.

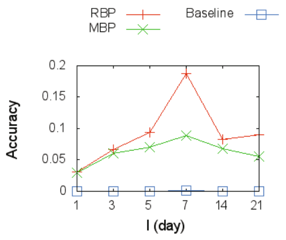
Impact of parameter *h* for RBP. Since the amount of reference history is a critical parameter for RBP algorithm, we evaluate the accuracy of recommendations under various reference histories. Figs. 6(a) and 6(b) depict the results of the App-small and App-large traces, and they reach the best accuracy on $h = 4$ and $h = 5$ respectively. Furthermore, the results of $h \geq 2$ are much better than the result of $h = 1$, because when $h = 1$, the number of reference points is too few to reflect the trend of users' usage. In addition, Fig. 6(c) shows the results of the Last.fm dataset, and the best accuracy falls on $h = 6$ and $l = 7$. Because the Last.fm is a sparse dataset, RBP algorithm needs more reference points and



(a) App-small dataset.

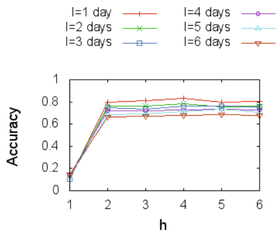


(b) App-large dataset.

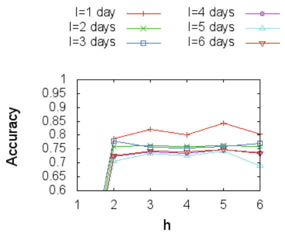


(c) Last.fm dataset.

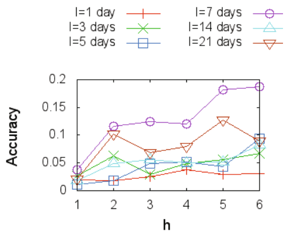
Fig. 5. Accuracy evaluation with the length of a time unit varied



(a) App-small dataset.



(b) App-large dataset.



(c) Last.fm dataset.

Fig. 6. Accuracy evaluation with the size of reference history varied

training data to achieve a better performance. Empirically, the setting of h does highly depend on different applications. In this paper, we suggest choosing a proper h larger than 4, since the regression line constructed in the first phase of RBP is more meaningful to reflect the trend of users' usage.

6 Conclusion

We proposed a novel dynamic preference prediction problem which is to dynamically quantize a user's preferences on Apps they have used from their usage traces. Two effective algorithms are designed to solve this problem. One is named Mode-base Prediction (MBP) which adopts a user's binary usage mode (active and inactive) and a proposed utility model to predict the preference value on an App. The other one is named Reference-base Prediction (RBP) which discovers a reference value by solving an optimization problem in a linear regression model and constructs a probabilistic model to check if the current behavior satisfies the reference model. RBP estimates the users' preferences by measuring the difference between actual usage and the derived reference value. In the experiments section, we evaluate the derived dynamic preferences by applying Collaborative Filtering. When the derived preferences can provide more accurate recommendation, the preferences are considered closer to users' actual affinities. As the experimental results show, the derived preferences of both MBP and RBP are effective. In addition, the RBP method can reach the accuracy of more than 80% for App traces. We suggest that the proposed dynamic preferences are valuable for many applications, such as providing recommendation of mobile applications, predicting and analysing users behavior, and make marketing decision.

Acknowledgement. Wen-Chih Peng was supported in part by the National Science Council, Project No.100-2218-E-009-016-MY3 and 100-2218-E-009-013-MY3, by Taiwan MoE ATU Program, by HTC and by Delta.

References

1. Dong, Y., Ke, Q., Rao, J., Wang, B., Wu, B.: Random walk based resource allocation: Predicting and recommending links in cross-operator mobile communication networks. In: 2011 IEEE 11th International Conference on Data Mining Workshops, ICDMW, Vancouver, BC, Canada, December 11, pp. 358–365 (2011)
2. Lymberopoulos, D., Zhao, P., König, A.C., Berberich, K., Liu, J.: Location-aware click prediction in mobile local search. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, pp. 413–422 (2011)
3. Guy, I., Zwerdling, N., Ronen, I., Carmel, D., Uziel, E.: Social media recommendation based on people and tags. In: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, pp. 194–201 (2010)
4. Yan, B., Chen, G.: Appjoy: personalized mobile application discovery. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys 2011, Bethesda, MD, USA, June 28-July 1, pp. 113–126 (2011)
5. Shi, K., Ali, K.: Getjar mobile application recommendations with very sparse datasets. In: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2012, Beijing, China, August 12-16, pp. 204–212 (2012)
6. Dror, G., Koenigstein, N., Koren, Y., Weimer, M.: The yahoo! music dataset and kdd-cup 2011. In: KDD-Cup Workshop (2011)
7. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28-July 1, pp. 447–456 (2009)
8. Leskovec, J., McGlohon, M., Faloutsos, C., Glance, N.S., Hurst, M.: Patterns of cascading behavior in large blog graphs. In: Proceedings of the Seventh SIAM International Conference on Data Mining, Minneapolis, Minnesota, USA, April 26-28 (2007)
9. Fei, H., Jiang, R., Yang, Y., Luo, B., Huan, J.: Content based social behavior prediction: a multi-task learning approach. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, pp. 995–1000 (2011)
10. Lathia, N., Hailes, S., Capra, L., Amatriain, X.: Temporal diversity in recommender systems. In: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, pp. 210–217 (2010)

11. Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J.: Temporal recommendation on graphs via long- and short-term preference fusion. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, pp. 723–732 (2010)
12. Celma, O.: Music Recommendation and Discovery in the Long Tail. Springer (2010)